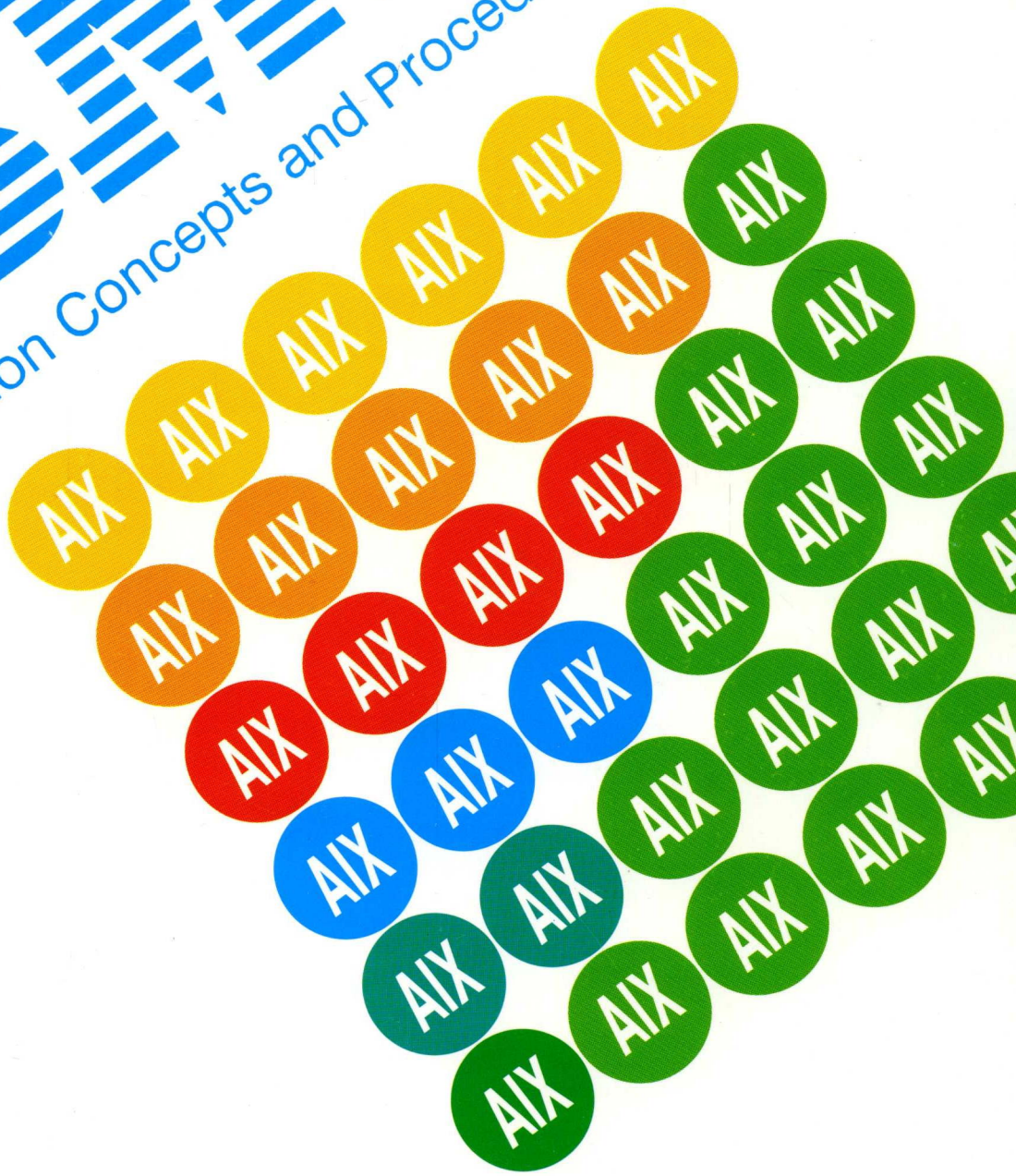
IBM

AIX Version 3 for
RISC System/6000™

Communication Concepts and Procedures

Volume 1



SC23-2203-00

First Edition (March 1990)

This edition of the *AIX Communication Concepts and Procedures for IBM RISC System/6000* applies to *Version Number 3* of the IBM AIX Base Operating System licensed program, AIX SNA Aervices/6000, AIX 3278/79 Emulation/6000, and AIX Network Management/6000 and to all subsequent releases of these products until otherwise indicated in new releases or technical newsletters.

The following paragraph does not apply to the United Kingdom or any country where such provisions are inconsistent with local law: INTERNATIONAL BUSINESS MACHINES CORPORATION PROVIDES THIS MANUAL "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESSED OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE. Some states do not allow disclaimer of express or implied warranties in certain transactions; therefore, this statement may not apply to you.

IBM does not warrant that the contents of this publication or the accompanying source code examples, whether individually or as one or more groups, will meet your requirements or that the publication or the accompanying source code examples are error-free.

This publication could include technical inaccuracies or typographical errors. Changes are periodically made to the information herein; these changes will be incorporated in new editions of the publication. IBM may make improvements and/or changes in the product(s) and/or the program(s) described in this publication at any time.

It is possible that this publication may contain reference to, or information about, IBM products (machines and programs), programming, or services that are not announced in your country. Such references or information must not be construed to mean that IBM intends to announce such IBM products, programming, or services in your country. Any reference to an IBM licensed program in this publication is not intended to state or imply that you can use only IBM's licensed program. You can use any functionally equivalent program instead.

Requests for copies of this publication and for technical information about IBM products should be made to your IBM Authorized Dealer or your IBM Marketing Representative.

A reader's comment form is provided at the back of this publication. If the form has been removed, address comments to IBM Corporation, Department 997, 11400 Burnet Road, Austin, Texas 78758–3493. IBM may use or distribute whatever information you supply in any way it believes appropriate without incurring any obligation to you.

The Network File System (NFS) was developed by Sun Microsystems, Inc.

The Rand MH Message Handling system developed by the Rand Corporation and the University of California.

This software and documentation is based in part on the Fourth Berkeley Software Distribution under license from The Regents of the University of California. We acknowledge the following institutions for their role in its development: the Electrical Engineering and Computer Sciences Department at the Berkeley Campus.

Trademarks and Acknowledgements

The following trademarks and acknowledgements apply to this book:

AIX is a trademark of International Business Machines Corporation.

AIXwindows is a trademark of International Business Machines Corporation.

Apollo is a trademark of Apollo Computer, Inc.

Hayes is a registered trademark of Hayes Microcomputer Products, Inc.

IBM is a registered trademark of International Business Machines Corporation.

NCK is a trademark of Apollo Computer, Inc.

NCS is a trademark of Apollo Computer, Inc.

Network Computing Kernel is a trademark of Apollo Computer, Inc.

Network Computing System is a trademark of Apollo Computer, Inc.

Network File System and NFS are trademarks of Sun Microsystems, Inc.

RISC System/6000 is a trademark of International Business Machines Corporation.

SNA 3270 is a trademark of International Business Machines Corporation.

UNIX was developed and licensed by AT&T and is a registered trademark of AT&T Corporation.

About This Book

This book contains conceptual and procedural information about a variety of communications facilities and applications.

Who Should Use This Book

This book is intended for people who want to use communications applications and services and who want to perform system management tasks that involve communication within a network.

How to Use This Book

The chapters of this book are divided into concepts for people who want to use the system or network, concepts for people who manage a system or network, and procedures for various tasks. The concept chapters are divided into overviews and subsequent information. The procedures are arranged in logical order and contain both using and system administration tasks.

Overview of Contents

The Communication Concepts and Procedures book contains information about several types of communication facilities. Part one contains information about emulators available for the RISC System/6000, including Asynchronous Terminal Emulation (ATE), 3278/79 Emulation (EM78), and the AIX 3270 Host Connection/6000 (HCON) program.

- Asynchronous Terminal Emulation (ATE) allows your system to emulate an asynchronous terminal attached to a remote computer. This chapter contains information about setting up ATE on your system and customizing it to meet your needs.
- The AIX 3278/79 Emulation/6000 Licensed Program (EM78) allows your system to emulate a terminal attached to an IBM System/370 host computer. This chapter contains information to help you install, configure, and customize EM78 on your system.
- The AIX 3270 Host Connection Program/6000 Licensed Program (HCON) allows your system to emulate a terminal or printer attached to an IBM System/370 host computer. This chapter contains information on installing HCON, setting up HCON users and profiles, and customizing HCON on your system. Also included is information to help you maintain HCON after it is in use.

Part two contains information about programs and facilities to use for network communications, including applications for remote terminal use (BNU), Mail and Message Handler facilities, applications for sharing files across a network (NFS/NIS and NCS), and applications for monitoring a network (Alerts and SNMP **xgmon**).

- The Basic Networking Utilities programs (BNU) are the AIX version of the UNIX-to-UNIX Copy Program (UUCP). They allow you to transfer files to and execute commands on remote systems, and to let remote systems transfer files to and execute command requests on your system. They also provide remote mail facilities. This chapter contains information to help you set up BNU and monitor its activities. It includes information on defining a BNU/UUCP network, scheduling access and contact times, and securing your system against unauthorized access by other systems that use BNU or UUCP.
- The Generic Data Link Control (GDLC) is a generic interface definition that allows both application and kernel users to have a common set of commands to control DLC device managers within the AIX Version 3 system. This chapter contains information on how to add, change, list, and remove data link controls.

- The Mail system is a general purpose, internetwork mail-routing facility. This system is not tied to any one transport protocol. It relays messages from one user to another across system and network domain boundaries. While processing the messages, the mail system can do a limited amount of message-header editing to put the message into a format that is appropriate for the receiving domain. This chapter provides information for both the daily use and management of Mail.
- The Message Handler (MH) package enables you to create, distribute, receive, view, process, and store messages. This chapter contains information to help you manage, customize, and use the MH package. Note that the MH package does not provide a message transport facility; instead, it relies on the transport facilities associated with the **sendmail** command.
- The Network Computing System (NCS) allows you to distribute processing tasks across resources in a network or internet by maintaining databases that control the information about the resources. NCS consists of three components: the Remote Procedure Call runtime library, the Location Broker, and the Network Interface Definition Language compiler. This chapter provides a brief introduction to the working of NCS and its components as well as detailed information on how to configure NCS.
- The Network File System (NFS) is a distributed file system that allows you to access files and directories remotely on a network as if they were local. The IBM NFS network information service (NIS) is a network service used to distribute system information on networked hosts. This chapter contains information to help you install, configure, and maintain NFS and NIS on your system.
- The AIX Network Management/6000 Licensed Program (Alerts program and the **xgmon** command) is a network management program for monitoring TCP/IP networks. It assists you in monitoring the status of all the machines on a network and helps you anticipate network problems. This chapter contains information about the Alerts monitoring application and how to start, use, maintain, and customize the **xgmon** program.

Part three contains information about network connectivity and includes Systems Network Architecture/6000, Transmission Control Protocol/Internet Protocol (TCP/IP), and X.25 Communications.

- TCP/IP is a communications subsystem that allows you to set up local area and wide area networks. TCP/IP allows you to transfer files between systems, log in to remote systems, run commands on and print files to remote systems, and communicate interactively or through mail with remote users. This chapter contains information to help you configure and customize TCP/IP. Also included is information to help you manage a network using TCP/IP. TCP/IP provides basic network management capability.
- Systems Network Architecture (SNA) is a specification that formally defines the functional responsibilities for components of a data communications system and specifies how those components must interact. This chapter contains information to help you to set up and customize SNA for your system. It includes information on SNA concepts, AIX SNA services, installing AIX SNA services, configuring a network using SNA, controlling SNA on the local system, and using the SNA extensions to the SMIT Interface.
- The X.25 communications service provides intercommunication between systems. It is particularly useful for communicating with people using different computer systems and for applications that access public data bases. This chapter includes information about planning, installing, configuring and customizing your X.25 network.

Highlighting

The following highlighting conventions are used in this book:

Bold	Identifies commands, keywords, files, directories, and other items whose names are predefined by the system.
Italics	Identifies parameters whose actual names or values are to be supplied by the user.
`Monospace`	Identifies examples of specific data values, examples of text similar to what you might see displayed, examples of portions of program code similar to what you might write as a programmer, messages from the system, or information you should actually type.

Related Publications

The following books contain information about or related to communications:

- *AIX Communications Programming Concepts for IBM RISC System/6000*, Order Number SC23–2206.
- *AIX General Concepts and Procedures for IBM RISC System/6000*, Order Number SC23–2202.
- *AIX Commands Reference for IBM RISC System/6000*, Order Number SC23–2199.
- *AIX Files Reference for IBM RISC System/6000*, Order Number SC23–2200.
- *IBM RISC System/6000 General Information and Planning Information Kit*, Order Number GK2T–0237.

Ordering Additional Copies of This Book

To order additional copies of this book, use Order Number SC23–2203.

Table of Contents

Part 1. Emulators

Chapter 1. Emulators Introduction

An emulator is an application that allows your machine to imitate another device. Emulators provide the functions of the device being emulated as if you were actually using that device. Two types of devices often emulated are terminals and printers. Depending on the features of the terminal emulator, it may connect to a host system and access host data, applications, and storage.

Some terminal emulators may provide a facility to transfer files to and from the host, and some may provide an application programming interface (API) to allow automation of host tasks. Depending on the features of the printer emulator, it may allow the host to either print files on a local printer or store the files in printable form on your machine so that it may be printed or edited later.

The following emulators are available:

- 3270 Host Connection Program (HCON) emulates a 3278/79 display (including models 2, 3, 4, 5, and 3270 base and extended attribute support) or a 3286/87 printer. It communicates with a System/370 host using the 3270 Connection Adapter in Distribution Function Terminal (DFT) mode or the System/370 Host Interface Adapter. HCON can establish simultaneous emulation sessions with one or more hosts including both display sessions and printer sessions. It also provides a file transfer facility and an Application Programming Interface (API). The **e789** command starts the HCON emulator. For more information, see the HCON Overview on page 4–1.
- 3278/79 Emulation (EM78) emulates a 3278/79 display. It communicates with a System/370 host using the 3270 Connection Adapter in Control Unit Terminal (CUT) mode. EM78 can establish one display session at a time. It also provides a file transfer facility. The **em78** command starts EM78. For more information, see the EM78 Overview on page 3–1.
- Asynchronous Terminal Emulation (ATE) allows your terminal to emulate a remote system, including another RISC System/6000 machine or an IBM PS/2 system. The **ate** command starts ATE. For more information, see the ATE Overview on page 2–1.
- The **telnet** command implements the TELNET Protocol, which allows remote login to other hosts. It uses the Transmission Control Protocol/Internet Protocol (TCP/IP) discussed on page 14–1 to communicate with other hosts in the network. See the **telnet, tn, tn3270** Command in *Commands Reference*.

Related Information

HCON Overview on page 4–1, EM78 Overview on page 3–1, ATE Overview on page 2–1.

The **e789** command, **em78** command, **ate** command, **telnet, tn, tn3270** command.

Chapter 2. Asynchronous Terminal Emulation (ATE)

The Asynchronous Terminal Emulation (ATE) program allows your terminal to emulate a remote system, such as another RISC System/6000 system or an IBM PS/2 system. This chapter provides information about using ATE, including descriptions of the menus and control keys. It also discusses management of ATE, including information about setting up connections between systems, setting up ports, creating a dialing directory file, and customizing ATE for individual systems or users.

ATE Overview

Asynchronous Terminal Emulation (ATE) allows your terminal to emulate a remote system. With ATE you can connect to most systems that support asynchronous terminals, including other RISC System/6000 systems and IBM PS/2 systems. ATE can cause the remote system to see your terminal either as your display or as a DEC VT100 terminal. The VT100 option allows you to log into systems that do not support your own terminal, but that do support VT100 terminals.

ATE uses both direct (cabled) and modem connections to communicate between your system and a remote system. Thus you can use ATE to connect either to a system in the next room or to one across the country. For a direct connection, you need to know the port to use on your system. For a modem connection, you need to know the port to use on your system and the telephone number of the remote system. You must also have a login and password on the remote system.

Using ATE, you can run commands on the remote system, send and receive files, and use the **xmodem** protocol to check the data integrity in the files you transfer between systems. You can also use a capture file to record, or *capture*, incoming data from the remote system.

ATE is menu driven and uses subcommands. When you issue the **ate** command to start ATE, it displays the Unconnected Main Menu. From this menu you can issue subcommands that allow you to

- Temporarily change characteristics of ATE (**modify**, **alter**).
- Connect to another system (**directory**, **connect**).
- Get help (**help**).
- Execute AIX commands on your system (**perform**).
- Leave ATE (**quit**).

Depending on the subcommand you issue at the Unconnected Main Menu, various submenus are displayed. When you use the **connect** or **directory** subcommand to connect to a remote system, ATE displays the Connected Main Menu. Using the **modify** and **alter** subcommands from one of the Main Menus, you can cause ATE to display the Modify Menu or the Alter Menu, respectively. When you run the **directory** subcommand, ATE displays your dialing directory (a list of phone numbers) in a columnar format.

From the Connected Main Menu, you can issue subcommands that allow you to

- Send files to and receive files from the remote system (**send**, **receive**).
- Send a break signal to the remote system (**break**).
- End the connection to the remote system (**terminate**).

In addition, the **modify**, **alter**, **help**, **perform**, and **quit** subcommands perform the same functions as those provided from the Unconnected Main Menu.

You can control certain actions of ATE with control key sequences. These key sequences are known as the CAPTURE_KEY, the MAINMENU_KEY, and the PREVIOUS_KEY. They are discussed in Using Control Keys with ATE. ATE is installed with default key combinations for these keys, but you can change the key combinations as you prefer by making entries in the ATE default file.

The ATE default file (**ate.def**) is created in your current directory the first time you run ATE. You can modify the file to customize various aspects of ATE. For example, you can change the name of your dialing directory file, the type of transfer protocols you use to send and receive files from the remote system, and the baud rate ATE expects your modem to use. For a complete discussion of ATE customization, see How to Customize the ATE Default File on page 2–9.

In addition to using the default file to customize ATE, you can temporarily change certain aspects of the way ATE functions by using the **modify** subcommand and the **alter** subcommand. Together, these subcommands can change most settings that you can change in the default file. The exceptions are the control key sequences, which can only be changed in the default file, and the name of your dialing directory, which can be changed with the **directory** subcommand. However, any changes made with the **modify**, **alter**, or **directory** subcommands are effective only for that session of ATE. The next time you run the program, the settings are those defined in your default file.

If you usually use a modem with ATE, you can create a dialing directory of up to 20 phone numbers. The **directory** subcommand then allows you to display the telephone numbers and select the one you need to connect to the system you are calling, instead of entering the phone number with the **connect** subcommand. By using a dialing directory, you can avoid having to look up the telephone number whenever you call a particular system. You can also specify certain data transmission characteristics in your dialing directory file. Specifying transmission characteristics in the dialing directory is useful if some of your connections use characteristics that differ from your defaults.

How to Create an ATE Dialing Directory File on page 2–10 explains how to set up a dialing directory. You can set up a personal directory; the person who administers your system can also set up a systemwide dialing directory. You can choose which dialing directory to use by making an entry in your ATE default file.

Using the ATE Unconnected Main Menu

The ATE Unconnected Main Menu is displayed any time you issue the **ate** command, provided that Asynchronous Terminal Emulation is installed. Once you have established a connection by issuing the ATE **connect** subcommand, the ATE Connected Main Menu is displayed.

The following subcommands can be issued from the ATE Unconnected Main Menu. To issue the subcommand, type its first letter at the command prompt on the menu. For example, type d to issue the **directory** subcommand.

alter	Temporarily changes data transmission characteristics, such as the speed of transmission for data.
connect	Makes a connection.
directory	Displays a dialing directory.
help	Displays help information.
modify	Temporarily modifies local settings, such as the capture file for incoming data.

perform Allows you to perform AIX commands within ATE.

quit Quits the ATE program.

The PREVIOUS_KEY control key can be used from the ATE Unconnected Main Menu.

Using the ATE Connected Main Menu

The ATE Connected Main Menu is displayed when you have established a connection using the ATE **connect** subcommand from the ATE Unconnected Main Menu, or when you press the MAINMENU_KEY key combination while connected to a remote system.

The following subcommands can be issued from the ATE Connected Main Menu. To issue the subcommand, type its first letter at the command prompt on the menu. For example, type `a` to issue the **alter** subcommand.

alter Temporarily changes data transmission characteristics, such as the speed of transmission for data.

break Sends a break signal to the remote system.

help Displays help information.

modify Temporarily modifies local settings used by the emulator, such as the capture file for incoming data.

perform Allows you to perform AIX commands within ATE.

quit Quits the ATE program.

receive Allows you to receive files from a remote system.

send Allows you to send files to a remote system.

terminate Terminates the ATE connection.

All three ATE control keys can be used from the ATE Connected Main Menu.

Using Control Keys with ATE

The following control keys can be used with the ATE emulator. You can change the key sequence for each function by editing the **ate.def** file.

CAPTURE_KEY Starts or stops saving data displayed on your screen during a connection. The default key sequence for the CAPTURE_KEY is Ctrl–B.

The CAPTURE_KEY has a switch or toggle effect. The first time you use it, you begin saving data. The next time you use it, you stop saving data. Data is saved in the capture file defined in the **ate.def** file.

The default capture file name is the **$HOME/kapture** file. You can use the **modify** subcommand to temporarily change the capture file name. Or, use the ATE default file to permanently change the name.

The CAPTURE_KEY key combination does not function while you are performing a file transfer operation, and is only valid when a connection has been established. If you press the CAPTURE_KEY key combination before a connection has been established, the next command you enter will fail and an error message will be displayed.

PREVIOUS_KEY	Returns you to the previously displayed screen. The actual screen you see depends on the screen in use when you pressed the key combination. The PREVIOUS_KEY can also be used to stop a file transfer operation. The default key sequence for the PREVIOUS_KEY is Ctrl–R. The PREVIOUS_KEY can be used from either ATE Main Menu.
MAINMENU_KEY	Displays the Connected Main Menu so you can issue an ATE subcommand. The default key sequence for the MAINMENU_KEY is Ctrl–V. Use this control key combination to display the Connected Main Menu after a connection to a remote system is established. If you press the MAINMENU_KEY key sequence before a connection has been established, the next command you enter will fail and an error message will be displayed.

By customizing the ATE default file, you can permanently change the control key settings and the capture file name.

Related Information

For more information on ATE menus, see the Modify Menu, discussed with the **modify** subcommand, and the Alter Menu, discussed with the **alter** subcommand. To display a dialing directory, use the **directory** subcommand.

The **ate** command starts ATE and displays the Unconnected Main Menu, from which you can access the ATE subcommands for connecting to remote systems and sending and receiving files.

The ATE Overview for System Management on page 2–5 explains how ATE works on your system.

How to Set up an ATE Dialing Directory on page 2–10 explains how to create a dialing directory and instruct ATE to use your dialing directory instead of the system default directory.

The **alter** subcommand and **modify** subcommand of the **ate** command temporarily change file transfer settings. How to Customize the ATE Default File on page 2–9 explains how to permanently change your file transfer settings, control keys, and dialing directory file.

The **xmodem** command allows you to transfer files using the **xmodem** protocol.

ATE Overview for System Management

The Asynchronous Terminal Emulation (ATE) program, an AIX Optional Program Product, allows your system to emulate a terminal on a remote system. Using ATE, you can log into most systems that support asynchronous terminals, including any system that supports RS232C or RS422A connections. Depending on a parameter the user sets, ATE causes the remote system to see the terminal as either an attached RISC System/6000 terminal or a DEC VT100 terminal.

This overview contains information about setting up ATE, customizing ATE, and the methods to use to change ATE characteristics. To install ATE, see How to Install AIX for RISC System/6000 Licensed Program Optional Program Products in *General Concepts and Procedures*.

Setting Up ATE

Before users on your system can run ATE, you must install the software and set up the ports and connections. ATE uses both direct (cabled) connections and modem connections. Local RS232C connections allow a maximum distance of 15 meters (50 feet) between machines, and RS422A connections allow up to 1200 meters (4000 feet). For more information about installing and setting up ATE, see How to Set Up ATE on page 2–8.

Before you can use ATE to call a remote system, make sure that the remote system's tty device is ready to accept a call. If users on remote systems use ATE to call your system, your tty device must be ready to accept a call.

Customizing ATE

After ATE is installed on your system, you may need to change its defaults. The first time ATE runs, it creates a default file (always named **ate.def**) in the current directory. By modifying this file, you can change the defaults to match those needed for your connection types and data transfers. Afterward, when you run ATE from that directory, it uses the settings in your customized **ate.def** file.

If you run ATE from a directory where no **ate.def** file exists, ATE creates a new **ate.def** file in that directory using the program defaults rather than those you have set. Thus, if you need to run ATE with different settings, you can keep appropriate **ate.def** files in different directories and run ATE from the appropriate directory depending on the settings you need for that session. On the other hand, running ATE from many directories results in multiple copies of the **ate.def** file cluttering up your system.

For more information, see How to Customize the ATE Default File on page 2–8.

If you need to change settings temporarily, you can use the **alter** or **modify** subcommand instead of modifying the default file. Settings changed with **alter** and **modify** remain in effect until you leave the ATE program with the **quit** subcommand. They then return to the defaults set in the **ate.def** file. Alternately, you can change settings for a specific modem connection using a dialing directory file entry. Settings changed in this way revert to the default when you terminate the connection, instead of when you leave ATE.

When installed, ATE uses a systemwide dialing directory file, the **/usr/lib/dir** file. Someone with root user authority can modify the **/usr/lib/dir** file to include numbers for modems used by everyone on the system. Individual users can also create their own dialing directory files and modify their copies of the **ate.def** file to cause ATE to use these directories. How to Set up an ATE Dialing Directory File on page 2–10 explains how to set up ATE to use a customized dialing directory.

If a number is not in the directory file, you can enter it with the **connect** subcommand. However, the dialing directory file lets you designate frequently used numbers by name. Additionally, a user can specify the baud rate, data character length, stop bits, parity, echoing, and linefeeds for a phone number if these characteristics differ from the defaults in the **ate.def** file.

Note: A dialing directory file can contain up to 20 lines (one entry per line). ATE ignores subsequent lines.

Changing ATE Characteristics

Following are the characteristics of ATE you can change and the appropriate methods for changing each. Note that all can be changed in the **ate.def** file.

Characteristic	Change with
Control keys	**ate.def** file only
Data character length	**alter** subcommand or dialing directory entry
Dialing directory file name	**directory** subcommand
Echoing (on or off)	**modify** subcommand or dialing directory entry
File name for incoming data (capture file)	**modify** subcommand
Final dial suffix for the modem	**alter** subcommand
Initial dial prefix for the modem	**alter** subcommand
Line feeds	**modify** subcommand or dialing directory entry
Number of redialing attempts	**alter** subcommand
Number of stop bits	**alter** subcommand or dialing directory entry
Parity (even or odd)	**alter** subcommand or dialing directory entry
Port name (device)	**alter** subcommand
Rate (bits per second)	**alter** subcommand or dialing directory entry
Telephone number	dialing directory entry
Transfer protocol (**pacing** or **xmodem**)	**alter** subcommand
Type of **pacing** (character or interval)	**alter** subcommand
VT100 emulation (on or off)	**modify** subcommand
Wait time between redialing attempts	**alter** subcommand
Write (capture) incoming data to a file	**modify** subcommand
Xon/Xoff protocol (on or off)	**modify** subcommand.

Related Information

To install ATE, see How to Install AIX for RISC System/6000 Licensed Program Optional Program Products in *General Concepts and Procedures*.

How to Set Up ATE on page 2–8 explains how to set up the connections necessary to run ATE.

The **alter** subcommand and **modify** subcommand of the **ate** command temporarily change file transfer settings. How to Customize the ATE Default File on page 2–8 explains how to permanently change your file transfer settings, control keys, and dialing directory file.

How to Set up an ATE Dialing Directory File on page 2–10 explains how to set up ATE to use a customized dialing directory.

Using ATE Control Keys on page 2–3 explains how the ATE control keys work.

For information on ATE menus, see the Unconnected Main Menu on page 2–2, the Connected Main Menu on page 2–3, the Modify Menu (discussed with the **modify** subcommand), and the Alter Menu (discussed with the **alter** subcommand). To display a dialing directory, use the **directory** subcommand.

The **ate** command starts ATE and displays the Unconnected Main Menu, from which you can access the ATE subcommands for changing characteristics, connecting to remote systems, and sending and receiving files.

List of Information about ATE System Management

The reference information about managing the ATE system falls into the categories of file formats and subcommands.

File Formats

You can create two files to customize ATE:

ate.def file format
: Sets default settings for connections.

Dialing directory file format
: Defines telephone numbers and settings for specific modem connections.

Subcommands of the ate command

The following subcommands of the **ate** command can be used to customize ATE:

alter subcommand
: Temporarily changes data transmission characteristics used on a connection.

modify subcommand
: Temporarily changes local settings for the emulator.

Related Information

The ATE Overview on page 2–1 introduces the ATE program.

How to Set Up ATE

Prerequisite Conditions

- The ATE program must be installed on your system. ATE is installed with the AIX Optional Program Products.
- You must have root user authority to set up the port for the communications device.

Procedure

To prepare Asynchronous Terminal Emulation (ATE) to run on your system, perform the following steps:

1. Install an asynchronous adapter card in an appropriate slot in the system unit, unless you are using a built–in serial port.
2. Plug the RS232C or RS422A cable into the adapter card or the built–in serial port.
3. Add a **ttydev** device description for the communications port. To do this, enter:

   ```
   smit mkdev
   ```

   Note: You can use the **mkdev** and **stty** commands instead of SMIT.

   Select the terminal type you want to emulate with ATE and make the necessary adjustments for the environment. The most common changes are line speed, parity settings, number of bits per character, and whether the line is to be driven as a remote or local line. Use `BPC 8` and `no parity` if NLS support is required.
4. Set up the port for the device. To set up a port to call out with ATE, use the **pdisable** command. For example, to set up port `tty1`, enter:

   ```
   pdisable tty1
   ```

   Note: To set up a port so that others can call *in*, use the **penable** command. For example, to let other systems call in to the `tty2` port, enter:

   ```
   penable tty2
   ```
5. Make sure the device has previously been defined to the remote system. Once the device is defined, the ATE emulation program must be customized to reflect the device settings on the remote system. Do this using the **alter** and **modify** subcommands, the **ate.def** default file, or a dialing directory file entry for the telephone connection.

Related Information

The **ate** command, **pdisable** command, **penable** command.

How to Install AIX for RISC System/6000 Licensed Program Optional Program Products in *General Concepts and Procedures.*

The **smit mkdev** command (described in the SMIT Overview in *General Concepts and Procedures*) or the **mkdev** command and **stty** command can be used to install and configure communications devices.

How to Customize the ATE Default File on page 2–9 and How to Set up an ATE Dialing Directory on page 2–10 explain procedures for customizing ATE. To temporarily change ATE settings, use the **alter** and **modify** subcommands.

ATE Overview for System Management on page 2–5 discusses tasks involved in managing ATE.

How to Edit the ATE Default File

Prerequisite Condition

- The ATE program must be set up on your system.

Procedure

When you invoke the ATE program, it searches your current directory for a copy of the **ate.def** file. If there is no **ate.def** file in the current directory, the ATE program creates one using default values. Edit this file to change the defaults. Afterwards when you run the **ate** command from that directory, the ATE defaults match those in the **ate.def** file.

To change the settings in your **ate.def** file, perform the following steps:

1. Start an ASCII text editor and display the **ate.def** file.
2. Delete all parameters that you do *not* want to change.

   This simplifies reading the file, since the system ignores variables that remain the same as the system's defaults.
3. Change the values for those variables you want to change.
4. Save the modified **ate.def** file.

The changes you make will take effect the next time you run ATE from the directory containing your customized **ate.def** file.

You can keep a copy of the **ate.def** file in any directory in which you have read and write permissions. For example, if you frequently need to run the ATE program with different defaults at different times, you can keep multiple copies of the **ate.def** file, with the appropriate settings, in different subdirectories of your **$HOME** directory. As an alternative, you can temporarily change most settings with the ATE **alter** subcommand and **modify** subcommand, or use a dialing directory entry to change settings for an individual modem connection.

Related Information

How to Set Up ATE on page 2–8.

For a listing of ATE settings that you can change, see the discussion of the **ate.def** file format.

To invoke the ATE program, use the **ate** command.

To temporarily change ATE default settings, use the **alter** subcommand and **modify** subcommand. To change ATE defaults for one modem connection while keeping other settings the same, see How to Set up an ATE Dialing Directory on page 2–10.

ATE Overview for System Management on page 2–5 discusses tasks involved in managing ATE and lists the aspects of ATE that can be customized.

How to Set up an ATE Dialing Directory

Prerequisite Conditions

- The ATE program must be set up on your system.
- To set up a systemwide dialing directory, you must have write access to the **/usr/lib/dir** file.

Procedure

You can create an ATE dialing directory file in any AIX directory to which you have both read and write access. Do the following to set up the dialing directory file for use:

1. Create the dialing directory file:
   a. Change directories to the AIX directory where you wish to put your dialing directory file.
   b. Using an ASCII text editor, open a file named with any valid AIX file name.
   c. Make telephone number entries using the format given in the dialing directory file format. For an example, look at the **/usr/lib/dir** file. Or, you can copy this file to use as a template.
   d. Save the file.

      Note: If your new directory is to be the systemwide default directory, save it with the name **/usr/lib/dir**.
2. If you named your file with any name other than the default (**/usr/lib/dir**), edit the **ate.def** file in the directory from which you run the ATE program. Change the `DIRECTORY` parameter to tell the program to use your new dialing directory file.
3. Start ATE and view your dialing directory with the **directory** subcommand.

Related Information

How to Set Up ATE on page 2–8, How to Edit the ATE Default File on page 2–9.

The ATE dialing directory file format and the ATE default (**ate.def**) file format.

The **ate** command starts ATE. The ATE **directory** subcommand and **connect** subcommand use the dialing directory file.

ATE Overview on page 2–1 introduces the ATE program.

ATE Overview for System Management on page 2–5 discusses tasks involved in managing ATE and lists the aspects of ATE that can be customized.

Chapter 3. AIX 3278/79 Emulation/6000 Program

The AIX 3278/79 Emulator (EM78) allows your system act like a terminal on a 3270 host system. This chapter contains both conceptual and procedural information about using and managing the 3278/79 Emulator. Key translation and unsupported key information is in the latter portion of the chapter.

EM78 Overview

EM78 is a 3278/79 terminal emulator that makes your system act like a terminal on a 3270 host system. The emulator must be installed on your system. Using EM78, you can run commands send files, or receive files on either a MVS/TSO host session or a VM/CMS host session. Issue the **em78** command to start the EM78 emulator.

The emulator acts as if you had just turned on the 3278/79 terminal. After you log in to the host, you can run commands and programs. For example, you might check for mail on the host. You can also upload and download files to and from the host as well as change the format of the data in the files as you transfer them. For example, you can translate files from ASCII to EBCDIC or add or remove carriage–return characters.

You can customize the keyboard mapping, color, and field attributes for the EM78 emulator. To customize the emulator you must edit a file following the EM78 Configuration file format and then install the changes with the **emkey** command.

Related Information

EM78 Overview for System Management on page 3–26.
How to Install EM78 on page 3–41, How to Customize EM78 on page 3–44.
The **em78** command, which starts EM78, the **emkey** command translates the ASCII profile into binary, **emsend (VM/CMS)** command, **emsend (MVS/TSO)** command, **emrcv (VM/CMS)** command, **emrcv (MVS/TSO)** command.

3278/79 Emulation Overview

The 3278/79 Emulation program emulates an IBM 3278 or 3279 display attached to an IBM System/370 computer. An emulator is an application that allows your machine to imitate another device. Emulators provide the functions of the device being emulated as if you were actually using that device.

The 3278/79 Emulation program **em78** command is used to start the emulation session. At the beginning of an emulation session, the emulator acts as if you had just turned on the 3278/79 terminal. After you log in to the System/370 host, you can run commands and programs. For example, you might check for mail on the host. You can also upload and download files to and from the host. This process is called file transfer.

The following information is provided to assist you in using the commands of the 3278/79 Emulation:

Using ASCII to EBCDIC Translation During File Transfer

Files stored on your workstation are in ASCII form, while files stored on a host system are in EBCDIC form. If you send an unsupported ASCII character such as a form feed to the host, a substitute character will replace the form feed in your file. When you copy the file back from EBCDIC to ASCII (from the host to your workstation), the ASCII equivalent of the substitute character will appear, not the form feed that you originally typed.

When a supported character is translated from EBCDIC to ASCII (from the host to your workstation), and then back to EBCDIC (the host), there is no overall change.

In addition to US English, the 3278/79 Emulation program supports 16 national languages. Due to different character representations, the translation charts are different for each language. The following translation charts are available:

In the translation tables, a substitute value of 1A for EBCDIC and 3F for ASCII is used when the character being sent is not supported. Some applications may display a substitute character, but other applications may not display anything.

Using Code Pages

A code page is a chart of characters and their numeric equivalents, used in file transfer. Each language supported has a code page associated with it. In the following list, CP represents code page, and CECP represents country–extended code page:

National Language	Code Page (CP or CECP)
Austrian/German	CECP 273
Belgian	CP 500
Canadian French	CECP 037
Danish	CECP 277
UK English	CECP 285
US English (ASCII)	
US English (TEXT) C	ECP 037 (modified)
Finnish	CECP 278
French (AZERTY)	CECP 297
Italian	CECP 280
Japanese English	CECP 281
Norwegian	CECP 277
Portuguese	CECP 282
Spanish–speaking	CECP 284
Swedish	CECP 278
Swiss–French	CECP 500
Swiss–German	CECP 500

3278/79 Emulation Program Limitations

The 3278/79 Emulation program should be used with the default keyboard and keyboard mapping.

The following IBM 3270 base features and functions are not supported by the 3278/79 Emulation program:

- Alternate cursor*
- Cursor blink*
- Keyboard click*
- Numeric lock feature
- Security keylock
- Magnetic reader control and accessories
- Displayed screens with more than 1920 characters
- Monocase switch
- Selector light pen
- Video output
- Katakana
- APL/TEXT character set
- Keyboards other than the 87–key (or 88–key Japanese English) 3278/79 keyboard
- RPQs
- Extended data–stream and associated keys:
  - Programmed symbols
  - Extended highlighting
  - Seven–color
- Structured field and attribute processing
- IBM 3270 diagnostic reset dump
- Explicit partitions
- Graphics escape
- IBM 3274 response time monitor

Note:

The features marked with an asterisk (*) have defined keys, but are not implemented.

Using 3278/79 Emulation Special Keys

The following is a description of special **em78** keys. These keys include functions that appear on the 3278/79 keyboards, such as **PF1** and **ENTER**, and special emulator keys, such as **Fast Left** which moves the cursor.

Each special emulator key has a default key sequence. By using the **emkey** command, different key sequences may be assigned to a specific key function. The following are special 3278/79 Emulation keys.

ASCIItoggle This key function is used to enter or exit ASCII Substitution Mode.

Attn A function key that operates only when a **B** is displayed in the Readiness and System Connection section of the Operator Information Area (OIA). This key lets you get the attention of the program.

Clear This function key erases the display and signals the host that a clear action has occurred.

Cursor Select This key function allows you to select a selectable field for processing.

Dev Cncl This function key cancels a pending print request.

Dup This key function key sends a request for a duplication operation to the program. It allows you to fill in information that is the same for every document.

Erase EOF This function key erases the input field from the cursor to the end of the field. The cursor does not move.

Erase Input This key function erases all input fields and moves the cursor to the first input character position on the screen.

Fast Left This key function allows you to move your cursor to the previous field.

Fast Right This key function allows you to move your cursor to the next field.

Field Mark This function key indicates the end of a field in an unformatted display.

Ident This function key identifies the next two numbers as the printer ID code. The ID is displayed in the Operator Information Area (OIA).

Program access function keys that communicate with a program:

PA1

PA2

PA3

Program function keys. These keys are used on the host:

PF1

PF2

PF3

PF4

PF5

PF6

PF7

PF8

PF9

PF10

PF11

PF12

PF13

PF14

PF15

PF16

PF17

PF18

PF19

PF20

PF21

PF22

PF23

PF24

Power–on reset
This funtion is equivalent to turning on the power to the IBM 3278/79 display or switching the normal/test toggle switch to normal.

Print This function key sends data from the display station to the selected host printer.

Quit This key function signals that you want to terminate the emulator.

Reset This key function resets the keyboard when it becomes locked.

Screenrepl This key function is used to replace the contents of a file with the current screen image.

Screensave This key function is used to append the screen contents to a file.

Shell This key function suspends the emulator and invokes a sub–shell for executing shell commands.

Sys Req This key function is used in conjunction with the Operator Information Area (OIA). If an **A** appears in the Readiness and System Connection section, this key function signals a test request message. If a **B** appears in the Readiness and System Connection section, this key function can clear the display or switch the display between an application and the control program.

Test This key function allows you to run special problem determination functions.

Using the 3278/79 Emulation Default Keyboard

The 3278/79 Emulation program comes with a default keyboard layout. You can change the keyboard layout by making a copy of the default customization file and then modifying it.

A key can be pressed alone or in combination with another key. A key pressed alone produces the unshifted function. Other functions are possible if the Shift (hollow up arrow), the Control (Ctrl), the Alternate (Alt), or the Alternate Graphics (AltGr) key is pressed and held while another key is pressed.

The NOT symbol, pipe symbol (|), and cent symbol (¢) only display when you are not in ASCII substitution mode (EBCDIC mode).

The following chart shows the default shifted, control, and alternate functions of the keys.

Function	Press	Function	Press
Screensave	Ctrl – Print Screen	PF1	F1
Screenrepl	Ctrl – Scroll Lock	PF2	F2
Shell	Ctrl – C	PF3	F3
Quit	Ctrl – D	PF4	F4
Power–on reset	Ctrl – P	PF5	F5
ASCIItoggle	Ctrl – T	PF6	F6
Cursor Select	Ctrl – F9	PF7	F7
Fast Left	Alt – Left Arrow	PF8	F8
Fast Right	Alt – Right Arrow	PF9	F9
Erase Input	Alt – End	PF10	F10
Dup	Alt – Page Down	PF11	F11
Test	Alt – Scroll Lock	PF12	F12
Sys Req	Alt – Print Screen	PF13	Shift – F1
Reset	Ctrl – Esc	PF14	Shift – F2
PA1	Page Up	PF15	Shift – F3
PA2	Page Down	PF16	Shift – F4
Clear	Pause	PF17	Shift – F5
Print	Print Screen	PF18	Shift – F6
Attn	Esc	PF19	Shift – F7
Ident	Shift – Print Screen	PF20	Shift – F8
Dev Cncl	Shift – Scroll Lock	PF21	Shift – F9
Field Mark	Shift – Home	PF22	Shift – F10
PA3	Shift – Page Up	PF23	Shift – F11
Erase EOF	Shift – End	PF24	Shift – F12

Using the 3278/79 Emulation Alternate Keyboard

The 3278/79 Emulation program comes with an alternate keyboard layout. You can use this keyboard layout in place of the default keyboard layout.

A key can be pressed alone or in combination with another key. A key pressed alone produces the unshifted function. Other functions are possible if the Shift (hollow up arrow), the Control (Ctrl), the Alternate (Alt), or the Alternate Graphics (AltGr) key is pressed and held while another key is pressed.

The NOT symbol, pipe symbol (|), and cent symbol (¢) only display when you are not in ASCII substitution mode (EBCDIC mode).

The following chart shows the alternate shifted, control, and alternate functions of the keys:

Alternate Keyboard Layout			
Function	**Press**	**Function**	**Press**
Screensave	Ctrl – Print Screen	PF1	F1
Screenrepl	Ctrl – Scroll Lock	PF2	F2
Shell	Ctrl – C	PF3	F3
Quit	Ctrl – D	PF4	F4
Power–on reset	Ctrl – P	PF5	F5
ASCIItoggle	Ctrl – T	PF6	F6
Cursor Select	Ctrl – F9	PF7	F7
Fast Left	Alt – Left Arrow	PF8	F8
Fast Right	Alt – Right Arrow	PF9	F9
Erase Input	Alt – End	PF10	F10
Dup	Shift – Scroll Lock	PF11	F11
Test	Alt – Scroll Lock	PF12	F12
Sys Req	Alt – Print Screen	PF13	Shift – F1
Reset	Ctrl – Esc	PF14	Shift – F2
PA1	Alt – Insert	PF15	Shift – F3
PA2	Alt – Home	PF16	Shift – F4
Clear	Pause	PF17	Shift – F5
Print	Print Screen	PF18	Shift – F6
Attn	Esc	PF19	Shift – F7
Ident	Shift – Print Screen	PF20	Shift – F8
Dev Cncl	Shift – Scroll Lock	PF21	Shift – F9
Field Mark	Shift – Pause	PF22	Shift – F10
PA3	Shift – Page Up	PF23	Shift – F11
Erase EOF	Shift – End	PF24	Shift – F12

Unsupported Keys for National Languages

Some national language workstation keyboards have characters which are not supported on the host system. If you press any of the unsupported keys while in 3278/79 Emulation, you will hear a beep (unless you have sound turned off), and the character will not appear on your display. See Unsupported Characters on page 3–71.

The following national languages have unsupported characters:

Austrian/German
Belgian
Canadian–French
Danish
UK English
Finnish and Swedish
French (AZERTY)
Italian

Japanese English
Norwegian
Portuguese
Spanish–speaking
Swiss–French and Swiss–German

Using ASCII Substitution Mode

ASCII and EBCDIC are two common ways for computers to represent characters with numbers. For example, the computer might represent a letter B with the numeric value 66. The workstation uses ASCII, while most host systems use EBCDIC. EBCDIC uses different numeric values than ASCII. ASCII substitution mode is only valid if you are using a US English 3278/79 Emulation keyboard.

ASCII substitution mode translates the characters [,], ^, and ! so a host system that is receptive to ASCII mode can interpret the characters correctly. Without ASCII substitution mode, the host system interprets each of these four characters as an EBCDIC character and returns a different numeric value to your workstation. This new value produces a different character at your display than the character you expected to see.

To use ASCII substitution mode, request one of the following:

- The **asciitoggle** emulator command
- The **–a** flag with the **em78** command

The following chart shows the characters that display when the ASCII substitution mode is invoked, and when ASCII substitution mode is not invoked:

If you type this:	Displayed with ASCII Substitution mode:	Displayed without ASCII Substitution mode:
[(left bracket)	[	|
] (right bracket)	]	¢
^ (caret)	^	(NOT Sign)
! (exclamation point)	!	!

The exclamation point (!) is displayed the same for both ASCII mode (with ASCII substitution) and EBCDIC mode (without ASCII substitution), however, the ASCII value for ! is different from the EBCDIC value.

Understanding the 3278/79 Emulation Operator Information Area (OIA)

The 3278/79 Emulation Program Operator Information Area (OIA) appears on line 25 of the display. This information, generated by the control unit, only appears when you are connected to an active control unit.

The information is grouped into five categories:

- Readiness and System Connection
- Do Not Enter (Input Inhibited)
- Reminders
- Shifts and Modes
- Printer Status

The following chart shows where the different categories appear on the status line:

Readiness and System Connection	Do Not Enter (Input Inhibited)	Reminders	Shifts and Modes	Printer Status

Readiness and System Connection Information

The following table explains the values that can be displayed in the Readiness and System Connection area:

Displayed Symbol	Name	What This Means
4	Ready	The IBM 3274 Control Unit is ready.
I	Ready	The Display/Printer Adapter is ready.
i	Ready	The Work Station Adapter is ready.
S	Ready	The IBM 3174 Control Unit is ready.
A	Online	The control unit is connected to the host under rule A.
B	Online	The control unit is connected to the host under rule B.
"	MyJob	Your display station is working with your job (application).
*	System Operator	Your display station is connected to the system operator (control program).
?	Unowned	Your display station is connected to the host system but is not connected to your application program or the control program.
TEST	Test	Your display station is in test mode.

Do Not Enter (Input Inhibited) Information

The following table explains the values that can be displayed in the Do Not Enter (Input Inhibited) area.

Note: Letters in italics, such as *nnn*, represent a variable field. Numbers will replace the *nnn*.

Input Inhibited Displayable Values Part 1 of 2		
Displayed Symbol	**Name**	**What This Means**
X ( )	Terminal Wait	Time is required for the host system to perform a function. Wait.
X ? +	What?	The last operation was not accepted. Check to make sure the operation you want is correct, then press **Reset** and try again.
X SYSTEM	System lock	The host system locked your keyboard. Look for a message. Wait or press **Reset**.

Input Inhibited Displayable Values Part 2 of 2		
Displayed Symbol	**Name**	**What This Means**
X < 0 >	Go Elsewhere	You tried to enter, insert, erase, or delete a character when the cursor was in a protected area. Press **Reset** and correct the entry.
X 0 >	Too much Entered	You tried to insert more data than this field can hold. Press **Reset** and correct the entry.
X 0#?	What Number?	You entered an incorrect printer ID number. Press **Reset**.
X – f	Minus Function	The requested function is not available. Press **Reset**.
X /*nnn*	Machine Check	Your System Unit is not working properly. The error number *nnn* defines the type of Machine Check. Press **Reset** to restore your keyboard and continue. If the problem persists, follow local procedures for reporting problems.
X – + z_*nnn*	Communication Check	There is a problem with the communication line between the control unit and the host system. The error number *nnn* defines the type of Communication Check. Press **Reset** to restore your keyboard and continue. If the problem persists, follow local procedures for reporting problems.
X PROG*nnn*	Program Check	The control unit detected a programming error in the data it received from the host system. The error number *nnn* defines the type of Program Check. Press **Reset** to restore your keyboard and continue. If the problem persists, follow local procedures for reporting problems.
X = – = ( )	Printer Busy	The printer connected to your display is busy. If =–#*nn* is displayed in the printer status area at the far right of the status line, the printer is busy with your work. Wait for the print operation to finish or press **Dev Cncl** to cancel a pending print operation.
X = – = ( ) ( )	Printer Very Busy	This message is the same as Printer Busy except more time is anticipated before your operation is accepted.
X = – /	Printer Not Working	Your printer is not operating properly. Press **Dev Cncl** and select another printer.
X 0 X	Operator Unauthorized	You are not authorized to do that function. Press **Reset**.
X = < *	Operator Message Not Accepted	A message from the control operator was rejected. Press **Reset**.

Reminders

The following table explains the values that can be displayed in the Reminders area.

Note: Letters in italics, such as *nnn*, represent a variable field. Numbers will replace the *nnn*.

Displayed Symbol	Name	What This Means
– + z_*nnn*	Communication Error Reminder	The communication link between your control unit and the host system produces errors.

Shifts and Modes Information

The following table explains the values that can be displayed in the Shifts and Modes areas:

Displayed Symbol	Name	What This Means
|	Shift Mode	Your display station is in Shift Mode.
^	Insert Mode	Your display station is in Insert Mode. To turn off Insert Mode, press **Reset**.

Printer Status Information

The following table explains the values that can be displayed in the Printer Status area.

Note: Letters in italics, such as *nnn*, represent a variable field. Numbers will replace the *nnn*.

Displayed Symbol	Name	What This Means
= – = _ _	Assign Printer	You should replace the _ _ (underscores) with the ID of the printer you want to use. The two numbers you type are displayed in place of the underscores.
= – = *nn*	Printer Assignment	You are authorized to use the printer with ID *nn*.
= – #*nn*	Printer Printing	The printer with ID *nn* is printing your work.
= – ~*nn*	Printer Failure	The printer with ID *nn* stopped while printing your work.
= – = ??	Assignment Pending	Press the **Ident** key.
Blank		Printing cannot take place.

Changing Colors for Emulation on a Color Display

The customization file (**emdefs.p**, **emaltdefs.p** or a file that you have previously changed) contains information about the colors used for emulation on your display.

You can change colors two ways:

- Change the foreground and background colors for the four color definitions: COLOR1, COLOR2, COLOR3, and COLOR4.

  OR

- Assign new color definitions to IBM 3278/79 field attributes.

You can also change keyboard layout information at the same time you change the color definitions.

The default values for the foreground and background colors are:

Definition	Foreground	Background
COLOR1	cyan	black
COLOR2	ltgreen	black
COLOR3	red	black
COLOR4	white	black
Default	Uses foreground from COLOR1 definition.	Uses background from COLOR1 definition.
Status	Uses foreground from Default definition.	Uses background from Default definition.

Note that the system uses the default colors when no other color definition is provided. Also, the status line is sometimes called the Operator Information Area (OIA).

You must define the default and status line colors in terms of COLOR1, COLOR2, COLOR3, or COLOR4. For example, if you wanted to change the default color to a red foreground on a black background, you would change the `default` entry to `COLOR3`. However, if you wanted to change the default color to magenta foreground with yellow background, you would have to redefine one of the existing colors – for example, COLOR4 – then change the `default` entry to `COLOR4`.

Acceptable color choices available are:

FOREGROUND

f_black	f_gray	f_yellow
f_red	f_ltred	f_brown
f_green	f_ltgreen	f_white
f_blue	f_ltblue	f_hiwhite
f_magenta	f_ltmagenta	
f_cyan	f_ltcyan	

BACKGROUND

b_black	b_blue
b_red	b_magenta
b_green	b_cyan
b_yellow	b_white

Related Information

The **em78** command,**emkey** command, **emsend (VM/CMS)** command, **emsend (MVS/TSO)** command, **emrcv (VM/CMS)** command, **emrcv (MVS/TSO)** command.

How to Transfer Files Between the Host and Your Workstation

Prerequisite Tasks or Conditions

1. 3278/79 Emulation Program installed on your system.

Procedure

Use the following steps to transfer files:

1. Make sure that the AIX Operating System prompt displays.
2. Invoke 3278/79 Emulation with the **em78** command, and specify the flags that you want.
3. Establish a host session. Be certain that you are logged on with the ID to which you want to transfer files.
4. Activate the emulation sub–shell with the shell key (Ctrl–C, unless you changed the default combination). All 3278/79 Emulation file transfer commands must be issued from the 3278/79 Emulation sub–shell, regardless of whether the file is sent or received.
5. Send or receive files as follows:
   a. To send a file from your workstation to the host, enter the **emsend (VM/CMS)** command, or **emsend (MVS/TSO)** command with the flags you want, or invoke a shell script to send the file.
   b. To receive a file from the host to your workstation, enter the **emrcv (VM/CMS)** command, or **emrcv (MVS/TSO)** command with the flags you want, or invoke a shell script to receive the file.
6. When the file transfer is complete, exit out of the emulation sub–shell (Ctrl–D, unless you changed the default combination).

Related Information

The **emsend (VM/CMS)** command, **emsend (MVS/TSO)** command, **emrcv (VM/CMS)** command, **emrcv (MVS/TSO)** command.

How to Create a Shell Script for File Transfers on page 3–15.

How to Create a Shell Script for File Transfers

Prerequisite Tasks or Conditions

1. 3278/79 Emulation Program installed on your system.

Procedure

To create a shell script for downloading files from the host to your workstation, perform the following steps:

1. Use an ASCII editor create a file. The name of the file can be anything you want it to be. For example, `down`.
2. Edit the file, and enter the **emrcv (VM/CMS)**, or **emrcv (MVS/TSO)** command options you want to perform. For example, to receive a VM/CMS host file with EBCDIC to ASCII conversion, enter the following line:

   ```
   emrcv -v -f$1 -V"$1 $2 a (TEXT"
   ```

   In the above example, `-v` specifies that you want file transfer information such as bytes transferred, to be displayed. `-f` specifies what you want the file to be named on the workstation. `-V"$1 $2 a` is the host information string, where `$1` represents the filename, `$2` represents the filetype, and `a` represents the filemode. `(TEXT"` specifies EBCDIC to ASCII conversion. `$1` and `$2` values are replaced with the names you enter.
3. Save the file.
4. Give the file execute permission.

Related Information

The **emsend (VM/CMS)** command, **emsend (MVS/TSO)** command, **emrcv (VM/CMS)** command, **emrcv (MVS/TSO)** command.

How to Invoke a Shell Script for File Transfers on page 3–16.

How to Invoke a Shell Script for File Transfers

Prerequisite Tasks or Conditions

1. 3278/79 Emulation Program installed on your system.
2. A shell script must exist.
3. A host session must be established.

Procedure

To invoke a shell script for downloading files from the host system to your workstation, perform the following steps:

1. Enter the emulation sub–shell:

   `Ctrl–C`

   If you changed your default key combinations, the above value may be different.

2. Enter the name of the shell script along with the name of the host file you want to download to your workstation. For example, if the shell script used to download files is called `down`, to receive host file `sample script` on your workstation, you would enter:

   ```
   down sample script
   ```

   The host file `sample script` will be received on your workstation as `sample`.

3. Exit from the emulation sub–shell:

   `Ctrl–D`

   If you changed your default key combinations, the above value may be different.

Related Information

How to Create a Shell Script for File Transfers on page 3–15.

How to Replace Data in a Screen Save File

Prerequisite Tasks or Conditions

1. 3278/79 Emulation Program installed on your system.

Procedure

To replace the contents of a file with the data on the screen:

1. Invoke the emulator with the **em78** command. If you want to specify a particular file, use the **–r***File* flag followed by the file name.
2. When you view a screen you want to save press the **screenrepl** key (default key combination is Ctrl + Scroll Lock). If you did not specify a file with the **–r***File* flag, the emulator creates a temporary file called **$HOME/emrep***xxxx*. (The system chooses random characters for the *xxxx*.) The first time you press the **screenrepl** key, a message containing the name of the temporary file appears on the screen.

Related Information

How to Replace and Append Data in the Same Screen Save File on page 3–18.

How to Print a Screen on page 3–19.

How to Replace and Append Data in the Same Screen Save File

Prerequisite Tasks or Conditions

1. 3278/79 Emulation Program installed on your system.

Procedure

If you want the flexibility of using the same file to append and replace data:

1. Invoke the emulator with the **em78** command and use both the **–r***File* and **–s***File* flags. The same file name must follow both flags:

   ```
   em78 –rfilename –sfilename
   ```

2. When you view a screen you want to save:

   Press the **screenrepl** key (default key combination is Ctrl + Scroll Lock) to replace the current content of the file with the screen data.

   OR

   Press the **screensave** key (default key combination is Ctrl + Print Screen) to append the screen data to the file.

Related Information

The **em78** command.

How to Replace Data in a Screen Save File on page 3–18.

How to Print a Screen on page 3–19.

How to Print a Screen

Prerequisite Tasks or Conditions

1. 3278/79 Emulation Program installed on your system.

Procedure

To print a screen immediately, save the screen data to a printer device. This suspends emulation until the screen is printed.

1. Invoke the emulator with the **em78** command and either the **–r***File* or the **–s***File* flag followed by the file name of the printer. For example:

   ```
   em78 –r/dev/lpn
   ```

   In the above example, you would replace *n* with a number (for example, `–r/dev/lp0`).

2. When you view a screen you want to print, press the **screenrepl** key (default key combination is Ctrl + Scroll Lock) if you used **–r***File* flag or the **screensave** key (default key combination is Ctrl + Print Screen) if you used **–s***File* flag. Pressing the **screensave** or **screenrepl** key starts the printing process.

Related Information

The **em78** command.

How to Replace Data in a Screen Save File on page 3–17.

How to Replace and Append Data in the Same Screen Save File on page 3–18.

How to Print a Stored File from the Shell

Prerequisite Tasks or Conditions

1. 3278/79 Emulation Program installed on your system.

Procedure

To print a file that you previously stored with the **screensave** or **screenrepl** command:

1. Press the **shell** key to enter the emulation sub–shell (default key combination Ctrl–C).
2. When the **emshell>** prompt appears on the display, enter:

   `print` *`filename`*

   Replace *`filename`* with the name of the file you want to print.
3. Press the **Enter** key. When the file is on the print queue, the **emshell>** prompt appears on the display again so you can issue another command.
4. To return to the emulator at this point, press Ctrl–D.

Related Information

The **em78** command.

How to Create a Shell Script for File Transfers on page 3–15.

How to Change Foreground and Background Colors

Prerequisite Tasks or Conditions

1. 3278/79 Emulation Program installed on your system.

Procedure

To change the colors used during emulation:

1. Edit a customization file. This can be the default file (**emdefs.p**), the alternate file (**emaltdefs.p**), or a file you have previously changed. It is advisable that you make a copy of the default or alternate file and use the copy for your changes.
2. Edit the section entitled DEFINED COLORS, and choose the colors you want to use from the list of valid color choices. Change the customization file to include the colors you want. Be sure to separate each foreground and background color in a definition with a plus sign (+). Include a blank space before and after the +. For example:

   ```
   COLOR1  f_cyan + b_black
   ```

3. Use the **emkey** command to create the new profile.

Related Information

The **em78** command, **emkey** command.

How to Change Field Attribute Colors on page 3–22.

How to Change Field Attribute Colors

Prerequisite Tasks or Conditions

1. 3278/79 Emulation Program installed on your system.

Procedure

Field attributes are a combination of the following:

- Unprotected (u) or protected (p)
- Alphanumeric (a) or numeric (n)
- Normal intensity (l), intensified (h), or non–display (i)
- Non–detect (x) or detect (d) with a light pen. If you do not have a light pen on your system or the host, the system ignores this attribute.

To change field attribute colors:

1. Make a chart similar to the following:

Attribute	Color
ualx	
uald	
uahd	
uaix	
unlx	
unld	
unhd	
unix	
palx	
pald	
pahd	
paix	
pnlx	
pnld	
pnhd	
pnix	

2. Write the new color definitions you want into the second column.
3. Your new chart should look similar to the following:

Attribute	Color
ualx	COLOR2
uald	COLOR2
uahd	COLOR3
uaix	default
unlx	COLOR2

unld	COLOR2
unhd	COLOR3
unix	default
palx	COLOR1
pald	COLOR1
pahd	COLOR4
paix	default
pnix	COLOR1
pnld	COLOR1
pnhd	COLOR4
pnix	default

4. Use your new chart to help you edit the customization file.
5. Use the **emkey** command with the appropriate flags to create the new profile.

Related Information

The **em78** command, **emkey** command.

The EM78 Customization File Format in *Files Reference*

How to Change Foreground and Background Colors on page 3–21, How to Change Field Attribute Modes on page 3–24.

How to Change Field Attribute Modes

Prerequisite Tasks or Conditions

1. 3278/79 Emulation Program installed on your system.

Procedure

Field attribute modes can be one of the following:

- blink
- bold
- revideo (reverse video)
- underline

If you select a mode that your display does not support, no change will take place.

To change field attribute modes:

1. Make a copy of the color definitions chart. Your chart should be similar to the following:

Attribute	Color
ualx	COLOR2
uald	COLOR2
uahd	COLOR3
uaix	default
unlx	COLOR2
unld	COLOR2
unhd	COLOR3
unix	default
palx	COLOR1
pald	COLOR1
pahd	COLOR4
paix	default
pnlx	COLOR1
pnld	COLOR1
pnhd	COLOR4
pnix	default

2. Mark changes to the attribute modes on this chart.
3. To change attribute uahd to be bold, place a plus sign (+) after the color definition for this attribute and add the name of the mode. A blank space must be used before and after the plus sign (+):

   ```
   uahd COLOR3 + bold
   ```

   You can change the mode of an attribute without changing the color definition.

4. After you have made the mode change for uahd, your color definition chart should be similar to the following :

Attribute	Color
ualx	COLOR2
uald	COLOR2
uahd	COLOR3 + bold
uaix	default
unlx	COLOR2
unld	COLOR2
unhd	COLOR3
unix	default
palx	COLOR1
pald	COLOR1
pahd	COLOR4
paix	default
pnix	COLOR1
pnld	COLOR1
pnhd	COLOR4
pnix	default

5. Use your newly created chart to help you edit the customization file.
6. After you have finished editing the customization file, using the **emkey** command and the appropriate flags to create the new profile.

Related Information

The **em78** command, **emkey** command.

The EM78 Customization File Format in *Files Reference*

How to Change Foreground and Background Colors on page 3–21, How to Change Field Attribute Modes on page 3–24.

EM78 Overview for System Management

EM78 is a 3278/79 terminal emulator that makes your system act like a terminal on a 3270 host system. The emulator must be installed and configured on your system.

When you install EM78, the files necessary to configure EM78 are copied from an installation medium to a machine's hard disk. There are no executable 3278/79 emulation files on the machine when installation completes.

When you configure EM78, the executable 3278/79 emulation files are created. These executable files are the **em78** command, the **emsend** command, and the **emrcv** command. The **emconfig** command creates these files.

When you configure EM78, you must select from a menu the appropriate keyboard for your country. The file transfer programs are configured according to your selection of country from the menu by means of country-specific file transfer translate tables. Configuration is performed only once after EM78 is installed.

You can customize the keyboard mapping, color, and field attributes. To customize the EM78 emulator you must edit a file that follows the EM78 Customization file format and install the changes with the **emkey** command.

For further information read Customizing EM78 on page 3–26, How to Install EM78 on page 3–41 and How to Configure EM78 on page 3–43.

Related Information

The **em78** command, **emconfig** command, **emkey** command, **emrcv** command, and **emsend** command.

Customizing EM78

EM78 is installed already customized. The default customization file is the **emdefs.p** file, which contains settings for key definitions, screen colors, and field attribute colors and modes.

The **emaltdefs.p** file is the alternate customization file that comes with EM78. It has different settings and is already in the EM78 Customization file format. You can install this file using the the **emkey** command.

To customize your EM78 emulator, you can:

- Copy and edit one of the existing customization files and install the changes using the **emkey** command. You should copy the file and edit the copy according to the EM78 Customization file format.
- Create and customize a file and install the changes using the **emkey** command. The file must follow the EM78 Customization file format.
- If you are familiar with the EM78 Customization file format, run the **emkey** command using standard input instead of specifying a file. This method is only for temporary changes, fast changes, or few changes. No record of the changes is made.

Some interfaces such as X-Windows override this configuration. If the appearance of EM78 does not match the way it is customized and you are running EM78 with an interface such as X-Windows, then run the **em78** command in its own shell.

Related Information

The **emkey** command.

The **emdefs.p** file, **emaltdefs.p** file,EM78 Customization file.

EM78 Emulator Color Names

The following colors can be used by the EM78 Emulation program. Use only the colors listed when defining colors to be used during emulation.

Foreground colors are indicated by **f_** followed by the color name. Background colors are indicated by **b_** followed by the color name.

Foreground Colors

f_black	**f_gray**	**f_yellow**
f_red	**f_ltred**	**f_brown**
f_green	**f_ltgreen**	**f_white**
f_blue	**f_ltblue**	**f_hiwhite**
f_magenta	**f_ltmagenta**	
f_cyan	**f_ltcyan**	

Background Colors

b_black	**b_blue**
b_red	**b_magenta**
b_green	**b_cyan**
b_yellow	**b_white**

Related Information

EM78 Functions on page 3–27, How to Customize EM78 on page 3–44, Mapping Keys for EM78 on page 3–28.

EM78 Functions

The EM78 Emulation program defines certain functions and allows them to be mapped to workstation keys. These functions are not traditional 3278/79 functions; they are functions of the emulator. The following list contains all of the functions recognized by EM78. Use only the names listed to map workstation keys to functions in the EM78 customization file.

asciitoggle Toggles ASCII substitution mode. The ASCII substitution mode is valid only for U.S. English keyboards. The default state of the ASCII substitution mode is off. Invoking the **em78** command with the **–a** flag starts the emulator with ASCII substitution mode on. (Default: Ctrl + T).

poreset Executes a power-on reset of the emulator. (Default: Ctrl + P).

quit Exits the emulator. (Default: Ctrl + D).

Note: If you reassign the Ctrl + D key combination, then you must assign the quit function to another key combination. Otherwise, you can not leave the emulator.

screenrepl Replaces the contents of the file that was selected with the **em78 –r** flag with the screen image. (Default: Ctrl + Scroll Lock).

screensave Appends the screen image to the file that was selected with the **em78 –s** flag. (Default: Ctrl + Print Screen).

shell Enters a subshell of the emulator. (Default: Ctrl + C).

Related Information

EM78 Customization file format in *Files Reference*, EM78 Screen Color Names on page 3–27, How to Customize EM78 on page 3–44, Mapping Keys for EM78 on page 3–28.

EM78 Field Attribute Codes

The **emdefs.p** file and the **emaltdefs.p** file contain default information about field attributes for EM78. A file you create in the EM78 customization format can also contain field attribute definitions. Use these codes when you define field attributes in the

The characteristics you can define for a field are combinaitons of the following:

u	Unprotected
p	Protected
a	Alphanumeric
n	Numeric
l	Normal intensity
h	Intensified
i	Nondisplay
x	Nondetect
d	Detect. If your system or host has no light pen, this attribute is ignored.

Related Information

EM78 Customization file format in *Files Reference*

How to Customize EM78 on page 3–44.

Mapping Keys for EM78

The EM78 Emulation program allows users to map, or describe, keyboard keys. You can use prefixes, suffixes, and numbers to represent key positions or special characters.

Mnemonic key names identify certain national language characters. The 3278/79 key names represent 3278/79 keys, and the workstation key names represent the workstation keys.

Related Information

EM78 Emulator Color Names on page 3–27, EM78 Functions on page 3–27, IBM 3278/79 Key Names on page 3–37, Mnemonic Key Names on page 3–29, Workstation Keyboard Layout on page 3–29.

How to Customize EM78 on page 3–44.

Mnemonic Key Names

Workstation Mnemonic Keys are used to identify certain national language characters that are produced by pressing a combination of 3278/79 keys. The following list contains all of the mnemonic key names recognized by the EM78 Emulation program. Use only the names listed to assign mnemonic key names in the EM78 customization file.

acute	grave	circumflex	umlaut	cedilla	tilde
a_acute	a_grave	a_circumflex	a_umlaut	c_cedilla	a_tilde
A_acute	A_grave	A_circumflex	A_umlaut	C_cedilla	A_tilde
e_acute	e_grave	e_circumflex	e_umlaut	overcircle	o_tilde
E_acute	E_grave	E_circumflex	E_umlaut	a_overcircle	O_tilde
i_acute	i_grave	i_circumflex	i_umlaut	A_overcircle	n_tilde
I_acute	I_grave	I_circumflex	I_umlaut		N_tilde
o_acute	o_grave	o_circumflex	o_umlaut		
O_acute	O_grave	O_circum-flex	O_umlaut		
u_acute	u_grave	u_circumflex	u_umlaut		
U_acute	U_grave	U_circumflex	U_umlaut		
			y_umlaut		
			Y_umlaut		

Related Information

EM78 Customization file format in *Files Reference*

EM78 Emulator Color Names on page 3–27, EM78 Functions on page 3–27, How to Customize EM78 on page 3–44.

Workstation Keyboard Layout

The EM78 Emulation program uses prefixes, suffixes, and numbers to represent key positions. The following notation is used for prefixes and is valid only for the 101-key U.S. keyboard:

k Represents a workstation key.

k32_ Represents an IBM 3278/79 key.

Suffixes show a second key that, when pressed at the same time as the named key, produces a different function. The key names use the following notation for suffixes:

_s Shift key.

_c Control (Ctrl) key.

_a Alternate (Alt) key.

No suffix Only one key is pressed.

Some characters cannot be represented by a key number (for example, non spacing character sequences) because more than one key must be pressed to generate the character. These key sequences have mnemonic representations.

Key Numbers Used by EM78

The numbers used by EM78 to represent each key on the workstation keyboard are as follows:

Number	Key
1	grave accent and tilde (‘ ~)
2	1
3	2
4	3
5	4
6	5
7	6
8	7
9	8
10	9
11	0
12	dash and underscore (– _)
13	equal and plus sign (= +)
15	Backspace
16	Tab key
17	q
18	w
19	e
20	r
21	t
22	y
23	u
24	i
25	o
26	p
27	left bracket ([)
28	right bracket (])

Number	Key
29	backslash and pipe (\ |)
30	Caps Lock
31	a
32	s
33	d
34	f
35	g
36	h
37	j
38	k
39	l
40	semicolon and colon (; :)
41	right single and double quotes ('")
43	Enter key
44	left Shift key
46	z
47	x
48	c
49	v
50	b
51	n
52	m
53	comma (,)
54	period (.)
55	forward slash and question mark (/ ?)
57	right Shift key
58	Ctrl key
60	left Alt key
61	space bar

Number	Key
62	right Alt key
64	Action key
75	Insert key
76	Delete key
79	left arrow key
80	Home key
81	End key
83	up arrow key
84	down arrow key
85	Page Up key
86	Page Down key
89	right arrow key
90	Num Lock key
91	keypad 7
92	keypad 4
93	keypad 1
95	keypad forward slash (/)
96	keypad 8
97	keypad 5
98	keypad 2
99	keypad 0
100	keypad asterisk (*)
101	keypad 9
102	keypad 6
103	keypad 3
104	keypad decimal (.)
105	keypad minus (–)
106	keypad plus (+)
108	keypad Enter key

Number	Key
110	Esc key
112	F1
113	F2
114	F3
115	F4
116	F5
117	F6
118	F7
119	F8
120	F9
121	F10
122	F11
123	F12
124	Print Screen key
125	Scroll Lock key
126	Pause key

Workstation Key Names for U.S. English Key Combinations

The following list contains the names EM78 uses for all of the recognized workstation keys for U.S. English. Use only the names listed to assign 3278/79 Emulation functions to workstation keys:

k1	k1_s	k1_c	k1_a
k2	k2_s	k2_c	k2_a
k3	k3_s	k3_c	k3_a
k4	k4_s	k4_c	k4_a
k5	k5_s	k5_c	k5_a
k6	k6_s	k6_c	k6_a
k7	k7_s	k7_c	k7_a
k8	k8_s	k8_c	k8_a
k9	k9_s	k9_c	k9_a
k10	k10_s	k10_c	k10_a

k11	k11_s	k11_c	k11_a
k12	k12_s	k12_c	k12_a
k13	k13_s	k13_c	k13_a
k15	k15_s	k15_c	k15_a
k16	k16_s	k16_c	k16_a
k17	k17_s	k17_c	k17_a
k18	k18_s	k18_c	k18_a
k19	k19_s	k19_c	k19_a
k20	k20_s	k20_c	k20_a
k21	k21_s	k21_c	k21_a
k22	k22_s	k22_c	k22_a
k23	k23_s	k23_c	k23_a
k24	k24_s	k24_c	k24_a
k25	k25_s	k25_c	k25_a
k26	k26_s	k26_c	k26_a
k27	k27_s	k27_c	k27_a
k28	k28_s	k28_c	k28_a
k29	k29_s	k29_c	k29_a
k31	k31_s	k31_c	k31_a
k32	k32_s	k32_c	k32_a
k33	k33_s	k33_c	k33_a
k34	k34_s	k34_c	k34_a
k35	k35_s	k35_c	k35_a
k36	k36_s	k36_c	k36_a
k37	k37_s	k37_c	k37_a
k38	k38_s	k38_c	k38_a
k39	k39_s	k39_c	k39_a
k40	k40_s	k40_c	k40_a
k41	k41_s	k41_c	k41_a
k43	k43_s	k43_c	k43_a

k46	k46_s	k46_c	k46_a
k47	k47_s	k47_c	k47_a
k48	k48_s	k48_c	k48_a
k49	k49_s	k49_c	k49_a
k50	k50_s	k50_c	k50_a
k51	k51_s	k51_c	k51_a
k52	k52_s	k52_c	k52_a
k53	k53_s	k53_c	k53_a
k54	k54_s	k54_c	k54_a
k55	k55_s	k55_c	k55_a
k61	k61_s	k61_c	k61_a
k64	—	—	—
k75	k75_s	k75_c	k75_a
k76	k76_s	k76_c	k76_a
k79	k79_s	k79_c	k79_a
k80	k80_s	k80_c	k80_a
k81	k81_s	k81_c	k81_a
k83	k83_s	k83_c	k83_a
k84	k84_s	k84_c	k84_a
k85	k85_s	k85_c	k85_a
k86	k86_s	k86_c	k86_a
k89	k89_s	k89_c	k89_a
—	—	k90_c	k90_a
k91	k91_s	k91_c	—
k92	k92_s	k92_c	—
k93	k93_s	k93_c	—
k95	k95_s	k95_c	k95_a
k96	k96_s	k96_c	—
k97	k97_s	k97_c	—
k98	k98_s	k98_c	—

k99	k99_s	k99_c	—
k100	k100_s	k100_c	k100_a
k101	k101_s	k101_c	—
k102	k102_s	k102_c	—
k103	k103_s	k103_c	—
k104	k104_s	k104_c	k104_a
k105	k105_s	k105_c	k105_a
k106	k106_s	k106_c	k106_a
k108	k108_s	k108_c	k108_a
k110	k110_s	k110_c	k110_a
k112	k112_s	k112_c	k112_a
k113	k113_s	k113_c	k113_a
k114	k114_s	k114_c	k114_a
k115	k115_s	k115_c	k115_a
k116	k116_s	k116_c	k116_a
k117	k117_s	k117_c	k117_a
k118	k118_s	k118_c	k118_a
k119	k119_s	k119_c	k119_a
k120	k120_s	k120_c	k120_a
k121	k121_s	k121_c	k121_a
k122	k122_s	k122_c	k122_a
k123	k123_s	k123_c	k123_a
k124	k124_s	k124_c	k124_a
k125	k125_s	k125_c	k125_a
k126	k126_s	k126_c	k126_a

Note: The combination Control + Alternate + *AnotherKey* is not valid.

Related Information

EM78 Emulator Color Names on page 3–27, EM78 Functions on page 3–27, How to Customize EM78 on page 3–44.

IBM 3278/79 Key Names

The IBM 3278/79 Emulation program uses prefixes, suffixes, and numbers to represent key positions. The following notation is used for prefixes:

k The number represents a workstation key.

k32_ The number represents an IBM 3278/79 key.

Suffixes show a second key that, when pressed at the same time as the named key, produces a different function. The key names use the following notation for suffixes:

_s Shift key.

_c Control (Ctrl) key.

_a Alternate (Alt) key.

No suffix Only one key is pressed.

Note: The Control key is valid only for workstation key combinations.

The following list contains the names of all the recognized 3278/79 keys. Use only these names to assign 3278/79 Emulation Program (EM78) functions to workstation keys.

k32_1	k32_1_s	k32_1_a
k32_2	k32_2_s	k32_2_a
k32_3	k32_3_s	k32_3_a
k32_4	k32_4_s	k32_4_a
k32_5	k32_5_s	k32_5_a
k32_6	k32_6_s	k32_6_a
k32_7	k32_7_s	k32_7_a
k32_8	k32_8_s	k32_8_a
k32_9	k32_9_s	k32_9_a
k32_10	k32_10_s	k32_10_a
k32_11	k32_11_s	k32_11_a
k32_12	k32_12_s	k32_12_a
k32_13	k32_13_s	k32_13_a
k32_14	k32_14_s	k32_14_a
k32_15	k32_15_s	k32_15_a
k32_16	k32_16_s	k32_16_a
k32_17	k32_17_s	k32_17_a
k32_18	k32_18_s	k32_18_a

k32_19	k32_19_s	k32_19_a
k32_20	k32_20_s	k32_20_a
k32_21	k32_21_s	k32_21_a
k32_22	k32_22_s	k32_22_a
k32_23	k32_23_s	k32_23_a
k32_24	k32_24_s	k32_24_a
k32_25	k32_25_s	k32_25_a
k32_26	k32_26_s	k32_26_a
k32_27	k32_27_s	k32_27_a
k32_28	k32_28_s	k32_28_a
k32_29	k32_29_s	k32_29_a
k32_30	k32_30_s	k32_30_a
k32_31	k32_31_s	k32_31_a
k32_32	k32_32_s	k32_32_a
k32_33	k32_33_s	k32_33_a
k32_34	k32_34_s	k32_34_a
k32_35	k32_35_s	k32_35_a
k32_36	k32_36_s	k32_36_a
k32_37	k32_37_s	k32_37_a
k32_38	k32_38_s	k32_38_a
k32_39	k32_39_s	k32_39_a
k32_40	k32_40_s	k32_40_a
k32_41	k32_41_s	k32_41_a
k32_42	k32_42_s	k32_42_a
k32_43	k32_43_s	k32_43_a
k32_44	k32_44_s	k32_44_a
k32_45	k32_45_s	k32_45_a
k32_46	k32_46_s	k32_46_a
k32_47	k32_47_s	k32_47_a
k32_48	k32_48_s	k32_48_a

k32_49	k32_49_s	k32_49_a
k32_50	k32_50_s	k_32_50_a
k32_51	k32_51_s	k32_51_a
k32_51A	k32_51A_s	k32_51A_a

Note: The k32_51A key is valid only on the Japanese-English keyboard.

k32_52	k32_52_s	k32_52_a
k32_53	k32_53_s	k32_53_a
k32_54	k32_54_s	k32_54_a
k32_55	k32_55_s	k32_55_a
k32_56	k32_56_s	k32_56_a
k32_57	k32_57_s	k32_57_a
k32_58	k32_58_s	k32_58_a
k32_59	k32_59_s	k32_59_a
k32_60	k32_60_s	k32_60_a
k32_61	k32_61_s	k32_61_a
k32_62	k32_62_s	k32_62_a
k32_63	k32_63_s	k32_63_a
k32_64	k32_64_s	k32_64_a
k32_65	k32_65_s	k32_65_a
k32_66	k32_66_s	k32_66_a
k32_67	k32_67_s	k32_67_a
k32_68	k32_68_s	k32_68_a
k32_69	k32_69_s	k32_69_a
k32_70	k32_70_s	k32_70_a
k32_71	k32_71_s	k32_71_a
k32_72	k32_72_s	k32_72_a
k32_74	k32_74_s	k32_74_a
k32_75	k32_75_s	k32_75_a
k32_76	k32_76_s	k32_76_a
k32_77	k32_77_s	k32_77_a

k32_78	k32_78_s	k32_78_a
k32_79	k32_79_s	k32_79_a
k32_80	k32_80_s	k32_80_a
k32_81	k32_81_s	k32_81_a
k32_82	k32_82_s	k32_82_a
k32_83	k32_83_s	k32_83_a
k32_84	k32_84_s	k32_84_a
k32_85	k32_85_s	k32_85_a
k32_86	k32_86_s	k32_86_a
k32_87	k32_87_s	k32_87_a
k32_88	k32_88_s	k32_88_a

Related Information

EM78 Emulator Color Names on page 3–27, EM78 Functions on page 3–27, How to Customize EM78 on page 3–44.

How to Install AIX 3278/79 Emulation/6000 Program (EM78)

You can install the AIX 3278/79 Emulation/6000 licensed program the following ways:

- From preloaded disk
- From tape
- From a network server
- From diskette.

The following sections describe how the licensed program is packaged, the system requirements for installation, and procedures for installing the licensed program.

How the AIX 3278/79 Emulation/6000 Licensed Program is Packaged

AIX 3278/79 Emulation/6000 licensed program consists of the following parts:

em78.obj Contains the 278/79 Emulator.

em78m*Language*.**msg**
Contains the messages and helps in the specified *Language* for this licensed program. If you install multiole languages for this product, be sure that you install the preferred language first.

The licensed program may also contain update files.

This licensed program contains support for either Double–Byte Character Set (DBCS) features, that is features that support Kanji (Japanese language), or National Language Support (NLS) features. The installation process automatically installs the appropriate support, depending on your AIX Base Operating System (BOS) Runtime installation.

Prerequisite Tasks or Conditions

- You must log in as root user.
- You must have installed the AIX Base Operating System (BOS) Runtime (part of the IBM AIX Base Operating System licensed program).
- You must have installed the Base Application Development Toolkit (part of the AIX Base Operating System licensed program) before you attempt to statically bind any AIX 3278/79 Emulation/6000 modules.
- You must install the **em78.obj** before **em78m***Language*.**msg**.
- To *run* EM78, you must install and configure the appropriate communications devices. However, you can *configure*, and *customize* EM78 either before or after installing and configuring the devices.

Procedure to Install for Preloaded Disk

1. Enter the following on an AIX command line:

   ```
   smit instupdt
   ```

   This command invokes the System Management Interface Tool (SMIT), which presents a menu driven environment for the installation process.

2. Follow the directions and answer the prompts in the SMIT Install menus. Select **em78.obj**, **em78m***Language*.**msg**, and any updates for the licensed program.

Messages display as each part successfully completes installing.

Procedure to Install from Tape

1. Insert the tape containing **em78.obj** into the tape drive.
2. Enter the following on an AIX command line:

   ```
   smit instupdt
   ```

3. Follow the directions and answer the prompts in the SMIT Install menus. Select **em78.obj**, **em78m***Language*.**msg**, and any updates for the licensed program.

Messages display as each part successfully completes installing.

Procedure to Install over a Network

1. Refer to How to Install over a Network in the *Installation Kit for IBM AIX Version 3 for RISC System/6000* for instructions on how to set up the server and how to down load files to the client.
2. Select Optional Program Products from the System Startup menu.
3. Follow the directions and answer the prompts in the SMIT Install menus. Select the files that contain **em78.obj**, **em78m***Language*.**msg**, and any updates for the 3278/79 Emulation/6000 program.

Messages display as each part successfully completes installing.

Procedure to Install from Diskette

1. Insert the first **em78.obj** diskette into the diskette drive.
2. Enter the following on an AIX command line:

   ```
   smit instupdt
   ```

   This command invokes the System Management Interface Tool (SMIT), which presents a menu driven environment for the installation process.
3. Follow the directions and answer the prompts in the SMIT Install menus. First install **em78.obj**, then install **em78m***Language*.**msg**, and finally apply the updates.

Messages display as each part successfully completes installing.

Related Information

The **installp** command, **updatep** command, **instupdt** command, and the **smit** command. For information about these commands, use the InfoExplorer information retrieval facility. This information is also included in the *AIX Commands Reference for IBM RISC System/6000*.

How to Configure EM78 on page 3–43, Customizing EM78 on page 3–26,

EM78 Overview for Managers on page 3–26.

Installation Instructions for IBM AIX Version 3 for RISC System/6000.

The System Manager Interface Tool (SMIT) Overview, in *General Concepts and Procedures*, Using SMIT in *General Concepts and Procedures*.

How to Configure EM78

Prerequisite Task or Condition

1. EM78 must be installed.

Procedure

1. Log in as a user in the system group.
2. After the prompt, type `emconfig` and press the Enter key.
3. From the menu that is displayed, choose the language for 3278/79 emulation and communications with your host computer, and press the Enter key.
4. Wait a few seconds until configuration is complete.

Related Information

The **emconfig** command.

The **emaltdefs.p** file, **emdefs.p** file.

EM78 Emulator Color Names on page 3–27, EM78 Functions on page 3–27, Mapping Keys for EM78 on page 3–28.

How to Install EM78 on page 3–41, EM78 Overview for Managers on page 3–26.

How to Customize EM78

Prerequisite Task or Condition

1. EM78 must be installed and configured.

Procedure

1. Copy the **emdefs.p** or the **emaltdefs.p** file to your home directory. The files are located in the **/usr/lib/em78** directory.
2. Edit the new file with a text editor. Follow the directions and make the changes to the color, keys, or functions that you would like to customize.
3. Use the settings specific to the EM78 program for the mapping key names, emulator colors, field attributes, or functions.
4. Create a new EM78 profile using the **emkey** command. You must specify the new file name following the command or the default **emdefs.p** file is used.
5. The **emkey** command automatically creates the **emkeys.o** profile, which the EM78 program reads to understand the new information.

Related Information

The **emkey** command, **emaltdefs.p** file, **emdefs.p** file.

EM78 Emulator Color Names on page 3–27, EM78 Field Attribute Codes on page 3–28, EM78 Functions on page 3–27, Mapping Keys for EM78 on page 3–28.

EM78 Overview for Managers on page 3–26, How to Install EM78 on page 3–41, How to Configure EM78 on page 3–43.

ASCII to EBCDIC Translation for Austrian/German

This chart is interpreted from left to right, and top to bottom. For example, if the first ASCII Digit is A, and the second ASCII Digit is F, you would find the A in the left column, and then locate the F in the top column. The point where the two columns meet would then be what the ASCII value would be translated to in EBCDIC representation. For example, the ASCII value AF would be translated to EBCDIC value 8B.

Note: All digits are hexadecimal.

	0	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**A**	**B**	**C**	**D**	**E**	**F**
0	00	01	02	03	37	2D	2E	2F	16	05	0A	0B	0C	0D	0E	0F
1	10	11	12	13	3C	3D	32	26	18	19	3F	27	3F	3F	3F	3F
2	40	4F	7F	7B	5B	6C	50	7D	4D	5D	5C	4E	6B	60	4B	61
3	F0	F1	F2	F3	F4	F5	F6	F7	F8	F9	7A	5E	4C	7E	6E	6F
4	B5	C1	C2	C3	C4	C5	C6	C7	C8	C9	D1	D2	D3	D4	D5	D6
5	D7	D8	D9	E2	E3	E4	E5	E6	E7	E8	E9	63	EC	FC	5F	6D
6	79	81	82	83	84	85	86	87	88	89	91	92	93	94	95	96
7	97	98	99	A2	A3	A4	A5	A6	A7	A8	A9	43	BB	DC	59	07
8	68	D0	51	42	C0	44	47	48	52	53	54	57	56	58	4A	67
9	71	9C	9E	CB	6A	CD	DB	DD	DF	E0	5A	70	B1	80	3F	B4
A	45	55	CE	DE	49	69	9A	9B	AB	AF	BA	B8	B7	AA	8A	8B
B	3F	3F	3F	3F	3F	65	62	64	3F	3F	3F	3F	3F	B0	B2	3F
C	3F	3F	3F	3F	3F	3F	46	66	3F	3F	3F	3F	3F	3F	3F	9F
D	8C	AC	72	73	74	DA	75	76	77	3F	3F	3F	3F	CC	78	3F
E	EE	A1	EB	ED	CF	EF	A0	8E	AE	FE	FB	FD	8D	AD	BC	BE
F	CA	8F	BF	B9	B6	7C	3F	9D	90	BD	B3	3F	FA	EA	3F	FF

EBCDIC to ASCII Translation for Austrian/German

This chart is interpreted from left to right, and top to bottom. For example, if the first EBCDIC Digit is A, and the second EBCDIC Digit is F, you would find the A in the left column, and then locate the F in the top column. The point where the two columns meet would then be what the EBCDIC value would be translated to in ASCII representation. For example, the EBCDIC value AF would be translated to ASCII value A9.

Note: All digits are hexadecimal.

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0	00	01	02	03	1A	09	0A	7F	1A	1A	0A	0B	0C	0D	0E	0F
1	10	11	12	13	1A	0A	08	1A	18	19	1A	1A	1A	1A	1A	1A
2	1A	1A	1A	1A	1A	0A	17	1B	1A	1A	1A	1A	1A	05	06	07
3	1A	1A	16	1A	1A	1A	1A	04	1A	1A	1A	1A	14	15	1A	1A
4	20	1A	83	7B	85	A0	C6	86	87	A4	8E	2E	3C	28	2B	21
5	26	82	88	89	8A	A1	8C	8B	8D	7E	9A	24	2A	29	3B	5E
6	2D	2F	B6	5B	B7	B5	C7	8F	80	A5	94	2C	25	5F	3E	3F
7	9B	90	D2	D3	D4	D6	D7	D8	DE	60	3A	23	F5	27	3D	22
8	9D	61	62	63	64	65	66	67	68	69	AE	AF	D0	EC	E7	F1
9	F8	6A	6B	6C	6D	6E	6F	70	71	72	A6	A7	91	F7	92	CF
A	E6	E1	73	74	75	76	77	78	79	7A	AD	A8	D1	ED	E8	A9
B	BD	9C	BE	FA	9F	40	F4	AC	AB	F3	AA	7C	EE	F9	EF	F2
C	84	41	42	43	44	45	46	47	48	49	F0	93	DD	95	A2	E4
D	81	4A	4B	4C	4D	4E	4F	50	51	52	D5	96	7D	97	A3	98
E	99	1A	53	54	55	56	57	58	59	5A	FD	E2	5C	E3	E0	E5
F	30	31	32	33	34	35	36	37	38	39	FC	EA	5D	EB	E9	FF

ASCII to EBCDIC Translation for Belgian, Swiss–French, Swiss–German

This chart is interpreted from left to right, and top to bottom. For example, if the first ASCII Digit is A, and the second ASCII Digit is F, you would find the A in the left column, and then locate the F in the top column. The point where the two columns meet would then be what the ASCII value would be translated to in EBCDIC representation. For example, the ASCII value AF would be translated to EBCDIC value 8B.

Note: All digits are hexadecimal.

	0	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**A**	**B**	**C**	**D**	**E**	**F**
0	00	01	02	03	37	2D	2E	2F	16	05	0A	0B	0C	0D	0E	0F
1	10	11	12	13	3C	3D	32	26	18	19	3F	27	3F	3F	3F	3F
2	40	4F	7F	7B	5B	6C	50	7D	4D	5D	5C	4E	6B	60	4B	61
3	F0	F1	F2	F3	F4	F5	F6	F7	F8	F9	7A	5E	4C	7E	6E	6F
4	7C	C1	C2	C3	C4	C5	C6	C7	C8	C9	D1	D2	D3	D4	D5	D6
5	D7	D8	D9	E2	E3	E4	E5	E6	E7	E8	E9	4A	E0	5A	5F	6D
6	79	81	82	83	84	85	86	87	88	89	91	92	93	94	95	96
7	97	98	99	A2	A3	A4	A5	A6	A7	A8	A9	C0	BB	D0	A1	07
8	68	DC	51	42	43	44	47	48	52	53	54	57	56	58	63	67
9	71	9C	9E	CB	CC	CD	DB	DD	DF	EC	FC	70	B1	80	3F	B4
A	45	55	CE	DE	49	69	9A	9B	AB	AF	BA	B8	B7	AA	8A	8B
B	3F	3F	3F	3F	3F	65	62	64	3F	3F	3F	3F	3F	B0	B2	3F
C	3F	3F	3F	3F	3F	3F	46	66	3F	3F	3F	3F	3F	3F	3F	9F
D	8C	AC	72	73	74	DA	75	76	77	3F	3F	3F	3F	6A	78	3F
E	EE	59	EB	ED	CF	EF	A0	8E	AE	FE	FB	FD	8D	AD	BC	BE
F	CA	8F	BF	B9	B6	B5	3F	9D	90	BD	B3	3F	FA	EA	3F	FF

EBCDIC to ASCII Translation for Belgian, Swiss–French, Swiss–German

This chart is interpreted from left to right, and top to bottom. For example, if the first EBCDIC Digit is A, and the second EBCDIC Digit is F, you would find the A in the left column, and then locate the F in the top column. The point where the two columns meet would then be what the EBCDIC value would be translated to in ASCII representation. For example, the EBCDIC value AF would be translated to ASCII value A9.

Note: All digits are hexadecimal.

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0	00	01	02	03	1A	09	0A	7F	1A	1A	0A	0B	0C	0D	0E	0F
1	10	11	12	13	1A	0A	08	1A	18	19	1A	1A	1A	1A	1A	1A
2	1A	1A	1A	1A	1A	0A	17	1B	1A	1A	1A	1A	1A	05	06	07
3	1A	1A	16	1A	1A	1A	1A	04	1A	1A	1A	1A	14	15	1A	1A
4	20	1A	83	84	85	A0	C6	86	87	A4	5B	2E	3C	28	2B	21
5	26	82	88	89	8A	A1	8C	8B	8D	E1	5D	24	2A	29	3B	5E
6	2D	2F	B6	8E	B7	B5	C7	8F	80	A5	DD	2C	25	5F	3E	3F
7	9B	90	D2	D3	D4	D6	D7	D8	DE	60	3A	23	40	27	3D	22
8	9D	61	62	63	64	65	66	67	68	69	AE	AF	D0	EC	E7	F1
9	FB	6A	6B	6C	6D	6E	6F	70	71	72	A6	A7	91	F7	92	CF
A	E6	7E	73	74	75	76	77	78	79	7A	AD	A8	D1	ED	E8	A9
B	BD	9C	BE	FA	9F	F5	F4	AC	AB	F3	AA	7C	EE	F9	EF	F2
C	7B	41	42	43	44	45	46	47	48	49	F0	93	94	95	A2	E4
D	7D	4A	4B	4C	4D	4E	4F	50	51	52	D5	96	81	97	A3	98
E	5C	1A	53	54	55	56	57	58	59	5A	FD	E2	99	E3	E0	E5
F	30	31	32	33	34	35	36	37	38	39	FC	EA	9A	EB	E9	FF

ASCII to EBCDIC Translation for Canadian-French

This chart is interpreted from left to right, and top to bottom. For example, if the first ASCII Digit is A, and the second ASCII Digit is F, you would find the A in the left column, and then locate the F in the top column. The point where the two columns meet would then be what the ASCII value would be translated to in EBCDIC representation. For example, the ASCII value AF would be translated to EBCDIC value 8B.

Note: All digits are hexadecimal.

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0	00	01	02	03	37	2D	2E	2F	16	05	0A	0B	0C	0D	0E	0F
1	10	11	12	13	3C	3D	32	26	18	19	3F	27	3F	3F	3F	3F
2	40	5A	7F	7B	5B	6C	50	7D	4D	5D	5C	4E	6B	60	4B	61
3	F0	F1	F2	F3	F4	F5	F6	F7	F8	F9	7A	5E	4C	7E	6E	6F
4	7C	C1	C2	C3	C4	C5	C6	C7	C8	C9	D1	D2	D3	D4	D5	D6
5	D7	D8	D9	E2	E3	E4	E5	E6	E7	E8	E9	BA	E0	BB	B0	6D
6	79	81	82	83	84	85	86	87	88	89	91	92	93	94	95	96
7	97	98	99	A2	A3	A4	A5	A6	A7	A8	A9	C0	4F	D0	A1	07
8	68	DC	51	42	43	44	47	48	52	53	54	57	56	58	63	67
9	71	9C	9E	CB	CC	CD	DB	DD	DF	EC	FC	70	B1	80	3F	B4
A	45	55	CE	DE	49	69	9A	9B	AB	AF	5F	B8	B7	AA	8A	8B
B	3F	3F	3F	3F	3F	65	62	64	3F	3F	3F	3F	3F	4A	B2	3F
C	3F	3F	3F	3F	3F	3F	46	66	3F	3F	3F	3F	3F	3F	3F	9F
D	8C	AC	72	73	74	DA	75	76	77	3F	3F	3F	3F	6A	78	3F
E	EE	59	EB	ED	CF	EF	A0	8E	AE	FE	FB	FD	8D	AD	BC	BE
F	CA	8F	BF	B9	B6	B5	3F	9D	90	BD	B3	3F	FA	EA	3F	FF

EBCDIC to ASCII Translation for Canadian-French

This chart is interpreted from left to right, and top to bottom. For example, if the first EBCDIC Digit is A, and the second EBCDIC Digit is F, you would find the A in the left column, and then locate the F in the top column. The point where the two columns meet would then be what the EBCDIC value would be translated to in ASCII representation. For example, the EBCDIC value AF would be translated to ASCII value A9.

Note: All digits are hexadecimal.

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0	00	01	02	03	1A	09	0A	7F	1A	1A	0A	0B	0C	0D	0E	0F
1	10	11	12	13	1A	0A	08	1A	18	19	1A	1A	1A	1A	1A	1A
2	1A	1A	1A	1A	1A	0A	17	1B	1A	1A	1A	1A	1A	05	06	07
3	1A	1A	16	1A	1A	1A	1A	04	1A	1A	1A	1A	14	15	1A	1A
4	20	1A	83	84	85	A0	C6	86	87	A4	BD	2E	3C	28	2B	7C
5	26	82	88	89	8A	A1	8C	8B	8D	E1	21	24	2A	29	3B	AA
6	2D	2F	B6	8E	B7	B5	C7	8F	80	A5	DD	2C	25	5F	3E	3F
7	9B	90	D2	D3	D4	D6	D7	D8	DE	60	3A	23	40	27	3D	22
8	9D	61	62	63	64	65	66	67	68	69	AE	AF	D0	EC	E7	F1
9	F8	6A	6B	6C	6D	6E	6F	70	71	72	A6	A7	91	F7	92	CF
A	E6	7E	73	74	75	76	77	78	79	7A	AD	A8	D1	ED	E8	A9
B	5E	9C	BE	FA	9F	F5	F4	AC	AB	F3	5B	5D	EE	F9	EF	F2
C	7B	41	42	43	44	45	46	47	48	49	F0	93	94	95	A2	E4
D	7D	4A	4B	4C	4D	4E	4F	50	51	52	D5	96	81	97	A3	98
E	5C	1A	53	54	55	56	57	58	59	5A	FD	E2	99	E3	E0	E5
F	30	31	32	33	34	35	36	37	38	39	FC	EA	9A	EB	E9	FF

ASCII to EBCDIC Translation for Danish and Norwegian

This chart is interpreted from left to right, and top to bottom. For example, if the first ASCII Digit is A, and the second ASCII Digit is F, you would find the A in the left column, and then locate the F in the top column. The point where the two columns meet would then be what the ASCII value would be translated to in EBCDIC representation. For example, the ASCII value AF would be translated to EBCDIC value 8B.

Note: All digits are hexadecimal.

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0	00	01	02	03	37	2D	2E	2F	16	05	0A	0B	0C	0D	0E	0F
1	10	11	12	13	3C	3D	32	26	18	19	3F	27	3F	3F	3F	3F
2	40	4F	7F	4A	67	6C	50	7D	4D	5D	5C	4E	6B	60	4B	61
3	F0	F1	F2	F3	F4	F5	F6	F7	F8	F9	7A	5E	4C	7E	6E	6F
4	80	C1	C2	C3	C4	C5	C6	C7	C8	C9	D1	D2	D3	D4	D5	D6
5	D7	D8	D9	E2	E3	E4	E5	E6	E7	E8	E9	9E	E0	9F	5F	6D
6	79	81	82	83	84	85	86	87	88	89	91	92	93	94	95	96
7	97	98	99	A2	A3	A4	A5	A6	A7	A8	A9	9C	BB	47	DC	07
8	68	A1	51	42	43	44	D0	48	52	53	54	57	56	58	63	5B
9	71	C0	7B	CB	CC	CD	DB	DD	DF	EC	FC	6A	B1	7C	3F	B4
A	45	55	CE	DE	49	69	9A	9B	AB	AF	BA	B8	B7	AA	8A	8B
B	3F	3F	3F	3F	3F	65	62	64	3F	3F	3F	3F	3F	B0	B2	3F
C	3F	3F	3F	3F	3F	3F	46	66	3F	3F	3F	3F	3F	3F	3F	5A
D	8C	AC	72	73	74	DA	75	76	77	3F	3F	3F	3F	70	78	3F
E	EE	59	EB	ED	CF	EF	A0	8E	AE	FE	FB	FD	8D	AD	BC	BE
F	CA	8F	BF	B9	B6	B5	3F	9D	90	BD	B3	3F	FA	EA	3F	FF

EBCDIC to ASCII Translation for Danish and Norwegian

This chart is interpreted from left to right, and top to bottom. For example, if the first EBCDIC Digit is A, and the second EBCDIC Digit is F, you would find the A in the left column, and then locate the F in the top column. The point where the two columns meet would then be what the EBCDIC value would be translated to in ASCII representation. For example, the EBCDIC value AF would be translated to ASCII value A9.

Note: All digits are hexadecimal.

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0	00	01	02	03	1A	09	0A	7F	1A	1A	0A	0B	0C	0D	0E	0F
1	10	11	12	13	1A	0A	08	1A	18	19	1A	1A	1A	1A	1A	1A
2	1A	1A	1A	1A	1A	0A	17	1B	1A	1A	1A	1A	1A	05	06	07
3	1A	1A	16	1A	1A	1A	1A	04	1A	1A	1A	1A	14	15	1A	1A
4	20	1A	83	84	85	A0	C6	7D	87	A4	23	2E	3C	28	2B	21
5	26	82	88	89	8A	A1	8C	8B	8D	E1	CF	8F	2A	29	3B	5E
6	2D	2F	B6	8E	B7	B5	C7	24	80	A5	9B	2C	25	5F	3E	3F
7	DD	90	D2	D3	D4	D6	D7	D8	DE	60	3A	92	9D	27	3D	22
8	40	61	62	63	64	65	66	67	68	69	AE	AF	D0	EC	E7	F1
9	F8	6A	6B	6C	6D	6E	6F	70	71	72	A6	A7	7B	F7	5B	5D
A	E6	81	73	74	75	76	77	78	79	7A	AD	A8	D1	ED	E8	A9
B	BD	9C	BE	FA	9F	F5	F4	AC	AB	F3	AA	7C	EE	F9	EF	F2
C	91	41	42	43	44	45	46	47	48	49	F0	93	94	95	A2	E4
D	86	4A	4B	4C	4D	4E	4F	50	51	52	D5	96	7E	97	A3	98
E	5C	1A	53	54	55	56	57	58	59	5A	FD	E2	99	E3	E0	E5
F	30	31	32	33	34	35	36	37	38	39	FC	EA	9A	EB	E9	FF

ASCII to EBCDIC Translation for Finnish and Swedish

This chart is interpreted from left to right, and top to bottom. For example, if the first ASCII Digit is A, and the second ASCII Digit is F, you would find the A in the left column, and then locate the F in the top column. The point where the two columns meet would then be what the ASCII value would be translated to in EBCDIC representation. For example, the ASCII value AF would be translated to EBCDIC value 8B.

Note: All digits are hexadecimal.

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0	00	01	02	03	37	2D	2E	2F	16	05	0A	0B	0C	0D	0E	0F
1	10	11	12	13	3C	3D	32	26	18	19	3F	27	3F	3F	3F	3F
2	40	4F	7F	63	67	6C	50	7D	4D	5D	5C	4E	6B	60	4B	61
3	F0	F1	F2	F3	F4	F5	F6	F7	F8	F9	7A	5E	4C	7E	6E	6F
4	EC	C1	C2	C3	C4	C5	C6	C7	C8	C9	D1	D2	D3	D4	D5	D6
5	D7	D8	D9	E2	E3	E4	E5	E6	E7	E8	E9	B5	71	9F	5F	6D
6	51	81	82	83	84	85	86	87	88	89	91	92	93	94	95	96
7	97	98	99	A2	A3	A4	A5	A6	A7	A8	A9	43	BB	47	DC	07
8	68	A1	79	42	C0	44	D0	48	52	53	54	57	56	58	7B	5B
9	E0	9C	9E	CB	6A	CD	DB	DD	DF	7C	FC	70	B1	80	3F	B4
A	45	55	CE	DE	49	69	9A	9B	AB	AF	BA	B8	B7	AA	8A	8B
B	3F	3F	3F	3F	3F	65	62	64	3F	3F	3F	3F	3F	3F	3F	3F
C	3F	3F	3F	3F	3F	3F	46	66	3F	3F	3F	3F	3F	3F	3F	5A
D	8C	AC	72	73	74	DA	75	76	77	3F	3F	3F	3F	CC	78	3F
E	EE	59	EB	ED	CF	EF	A0	8E	AE	FE	FB	FD	8D	AD	BC	BE
F	CA	8F	BF	B9	B6	4A	3F	9D	90	BD	B3	3F	FA	EA	3F	FF

EBCDIC to ASCII Translation for Finnish and Swedish

This chart is interpreted from left to right, and top to bottom. For example, if the first EBCDIC Digit is A, and the second EBCDIC Digit is F, you would find the A in the left column, and then locate the F in the top column. The point where the two columns meet would then be what the EBCDIC value would be translated to in ASCII representation. For example, the EBCDIC value AF would be translated to ASCII value A9.

Note: All digits are hexadecimal.

	0	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**A**	**B**	**C**	**D**	**E**	**F**
0	00	01	02	03	1A	09	0A	7F	1A	1A	0A	0B	0C	0D	0E	0F
1	10	11	12	13	1A	0A	08	1A	18	19	1A	1A	1A	1A	1A	1A
2	1A	1A	1A	1A	1A	0A	17	1B	1A	1A	1A	1A	1A	05	06	07
3	1A	1A	16	1A	1A	1A	1A	04	1A	1A	1A	1A	14	15	1A	1A
4	20	1A	83	7B	85	A0	C6	7D	87	A4	F5	2E	3C	28	2B	21
5	26	60	88	89	8A	A1	8C	8B	8D	E1	CF	8F	2A	29	3B	5E
6	2D	2F	B6	23	B7	B5	C7	24	80	A5	94	2C	25	5F	3E	3F
7	9B	5C	D2	D3	D4	D6	D7	D8	DE	82	3A	8E	99	27	3D	22
8	9D	61	62	63	64	65	66	67	68	69	AE	AF	D0	EC	E7	F1
9	F8	6A	6B	6C	6D	6E	6F	70	71	72	A6	A7	91	F7	92	5D
A	E6	81	73	74	75	76	77	78	79	7A	AD	A8	D1	ED	E8	A9
B	BD	9C	BE	FA	9F	5B	F4	AC	AB	F3	AA	7C	EE	F9	EF	F2
C	84	41	42	43	44	45	46	47	48	49	F0	93	DD	95	A2	E4
D	86	4A	4B	4C	4D	4E	4F	50	51	52	D5	96	7E	97	A3	98
E	90	1A	53	54	55	56	57	58	59	5A	FD	E2	40	E3	E0	E5
F	30	31	32	33	34	35	36	37	38	39	FC	EA	9A	EB	E9	FF

ASCII to EBCDIC Translation for French (AZERTY)

This chart is interpreted from left to right, and top to bottom. For example, if the first ASCII Digit is A, and the second ASCII Digit is F, you would find the A in the left column, and then locate the F in the top column. The point where the two columns meet would then be what the ASCII value would be translated to in EBCDIC representation. For example, the ASCII value AF would be translated to EBCDIC value 8B.

Note: All digits are hexadecimal.

	0	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**A**	**B**	**C**	**D**	**E**	**F**
0	00	01	02	03	37	2D	2E	2F	16	05	0A	0B	0C	0D	0E	0F
1	10	11	12	13	3C	3D	32	26	18	19	3F	27	3F	3F	3F	3F
2	40	4F	7F	B1	5B	6C	50	7D	4D	5D	5C	4E	6B	60	4B	61
3	F0	F1	F2	F3	F4	F5	F6	F7	F8	F9	7A	5E	4C	7E	6E	6F
4	44	C1	C2	C3	C4	C5	C6	C7	C8	C9	D1	D2	D3	D4	D5	D6
5	D7	D8	D9	E2	E3	E4	E5	E6	E7	E8	E9	90	48	B5	5F	6D
6	A0	81	82	83	84	85	86	87	88	89	91	92	93	94	95	96
7	97	98	99	A2	A3	A4	A5	A6	A7	A8	A9	51	BB	54	BD	07
8	68	DC	C0	42	43	7C	47	E0	52	53	D0	57	56	58	63	67
9	71	9C	9E	CB	CC	CD	DB	6A	DF	EC	FC	70	7B	80	3F	B4
A	45	55	CE	DE	49	69	9A	9B	AB	AF	BA	B8	B7	AA	8A	8B
B	3F	3F	3F	3F	3F	65	62	64	3F	3F	3F	3F	3F	B0	B2	3F
C	3F	3F	3F	3F	3F	3F	46	66	3F	3F	3F	3F	3F	3F	3F	9F
D	8C	AC	72	73	74	DA	75	76	77	3F	3F	3F	3F	DD	78	3F
E	EE	59	EB	ED	CF	EF	79	8E	AE	FE	FB	FD	8D	AD	BC	BE
F	CA	8F	BF	B9	B6	5A	3F	9D	4A	A1	B3	3F	FA	EA	3F	FF

EBCDIC to ASCII Translation for French (AZERTY)

This chart is interpreted from left to right, and top to bottom. For example, if the first EBCDIC Digit is A, and the second EBCDIC Digit is F, you would find the A in the left column, and then locate the F in the top column. The point where the two columns meet would then be what the EBCDIC value would be translated to in ASCII representation. For example, the ASCII value AF would be translated to EBCDIC value A9.

Note: All digits are hexadecimal.

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0	00	01	02	03	1A	09	0A	7F	1A	1A	0A	0B	0C	0D	0E	0F
1	10	11	12	13	1A	0A	08	1A	18	19	1A	1A	1A	1A	1A	1A
2	1A	1A	1A	1A	1A	0A	17	1B	1A	1A	1A	1A	1A	05	06	07
3	1A	1A	16	1A	1A	1A	1A	04	1A	1A	1A	1A	14	15	1A	1A
4	20	1A	83	84	40	A0	C6	86	5C	A4	F8	2E	3C	28	2B	21
5	26	7B	88	89	7D	A1	8C	8B	8D	E1	F5	24	2A	29	3B	5E
6	2D	2F	B6	8E	B7	B5	C7	8F	80	A5	97	2C	25	5F	3E	3F
7	9B	90	D2	D3	D4	D6	D7	D8	DE	E6	3A	9C	85	27	3D	22
8	9D	61	62	63	64	65	66	67	68	69	AE	AF	D0	EC	E7	F1
9	5B	6A	6B	6C	6D	6E	6F	70	71	72	A6	A7	91	F7	92	CF
A	60	F9	73	74	75	76	77	78	79	7A	AD	A8	D1	ED	E8	A9
B	BD	23	BE	FA	9F	5D	F4	AC	AB	F3	AA	7C	EE	7E	EF	F2
C	82	41	42	43	44	45	46	47	48	49	F0	93	94	95	A2	E4
D	8A	4A	4B	4C	4D	4E	4F	50	51	52	D5	96	81	DD	A3	98
E	87	1A	53	54	55	56	57	58	59	5A	FD	E2	99	E3	E0	E5
F	30	31	32	33	34	35	36	37	38	39	FC	EA	9A	EB	E9	FF

ASCII to EBCDIC Translation for Italian

This chart is interpreted from left to right, and top to bottom. For example, if the first ASCII Digit is A, and the second ASCII Digit is F, you would find the A in the left column, and then locate the F in the top column. The point where the two columns meet would then be what the ASCII value would be translated to in EBCDIC representation. For example, the ASCII value AF would be translated to EBCDIC value 8B.

Note: All digits are hexadecimal.

	0	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**A**	**B**	**C**	**D**	**E**	**F**
0	00	01	02	03	37	2D	2E	2F	16	05	0A	0B	0C	0D	0E	0F
1	10	11	12	13	3C	3D	32	26	18	19	3F	27	3F	3F	3F	3F
2	40	4F	7F	B1	5B	6C	50	7D	4D	5D	5C	4E	6B	60	4B	61
3	F0	F1	F2	F3	F4	F5	F6	F7	F8	F9	7A	5E	4C	7E	6E	6F
4	B5	C1	C2	C3	C4	C5	C6	C7	C8	C9	D1	D2	D3	D4	D5	D6
5	D7	D8	D9	E2	E3	E4	E5	E6	E7	E8	E9	90	48	51	5F	6D
6	DD	81	82	83	84	85	86	87	88	89	91	92	93	94	95	96
7	97	98	99	A2	A3	A4	A5	A6	A7	A8	A9	44	BB	54	58	07
8	68	DC	5A	42	43	C0	47	E0	52	53	D0	57	56	A1	63	67
9	71	9C	9E	CB	CC	6A	DB	79	DF	EC	FC	70	7B	80	3F	B4
A	45	55	CE	DE	49	69	9A	9B	AB	AF	BA	B8	B7	AA	8A	8B
B	3F	3F	3F	3F	3F	65	62	64	3F	3F	3F	3F	3F	B0	B2	3F
C	3F	3F	3F	3F	3F	3F	46	66	3F	3F	3F	3F	3F	3F	3F	9F
D	8C	AC	72	73	74	DA	75	76	77	3F	3F	3F	3F	CD	78	3F
E	EE	59	EB	ED	CF	EF	A0	8E	AE	FE	FB	FD	8D	AD	BC	BE
F	CA	8F	BF	B9	B6	7C	3F	9D	4A	BD	B3	3F	FA	EA	3F	FF

EBCDIC to ASCII Translation for Italian

This chart is interpreted from left to right, and top to bottom. For example, if the first EBCDIC Digit is A, and the second EBCDIC Digit is F, you would find the A in the left column, and then locate the F in the top column. The point where the two columns meet would then be what the EBCDIC value would be translated to in ASCII representation. For example, the EBCDIC value AF would be translated to ASCII value A9.

Note: All digits are hexadecimal.

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0	00	01	02	03	1A	09	0A	7F	1A	1A	0A	0B	0C	0D	0E	0F
1	10	11	12	13	1A	0A	08	1A	18	19	1A	1A	1A	1A	1A	1A
2	1A	1A	1A	1A	1A	0A	17	1B	1A	1A	1A	1A	1A	05	06	07
3	1A	1A	16	1A	1A	1A	1A	04	1A	1A	1A	1A	14	15	1A	1A
4	20	1A	83	84	7B	A0	C6	86	5C	A4	F8	2E	3C	28	2B	21
5	26	5D	88	89	7D	A1	8C	8B	7E	E1	82	24	2A	29	3B	5E
6	2D	2F	B6	8E	B7	B5	C7	8F	80	A5	95	2C	25	5F	3E	3F
7	9B	90	D2	D3	D4	D6	D7	D8	DE	97	3A	9C	F5	27	3D	22
8	9D	61	62	63	64	65	66	67	68	69	AE	AF	D0	EC	E7	F1
9	5B	6A	6B	6C	6D	6E	6F	70	71	72	A6	A7	91	F7	92	CF
A	E6	8D	73	74	75	76	77	78	79	7A	AD	A8	D1	ED	E8	A9
B	BD	23	BE	FA	9F	40	F4	AC	AB	F3	AA	7C	EE	F9	EF	F2
C	85	41	42	43	44	45	46	47	48	49	F0	93	94	DD	A2	E4
D	8A	4A	4B	4C	4D	4E	4F	50	51	52	D5	96	81	60	A3	98
E	87	1A	53	54	55	56	57	58	59	5A	FD	E2	99	E3	E0	E5
F	30	31	32	33	34	35	36	37	38	39	FC	EA	9A	EB	E9	FF

ASCII to EBCDIC Translation for Japanese English

This chart is interpreted from left to right, and top to bottom. For example, if the first ASCII Digit is A, and the second ASCII Digit is F, you would find the A in the left column, and then locate the F in the top column. The point where the two columns meet would then be what the ASCII value would be translated to in EBCDIC representation. For example, the ASCII value AF would be translated to EBCDIC value 8B.

Note: All digits are hexadecimal.

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0	00	01	02	03	37	2D	2E	2F	16	05	0A	0B	0C	0D	0E	0F
1	10	11	12	13	3C	3D	32	26	18	19	3F	27	3F	3F	3F	3F
2	40	5A	7F	7B	E0	6C	50	7D	4D	5D	5C	4E	6B	60	4B	61
3	F0	F1	F2	F3	F4	F5	F6	F7	F8	F9	7A	5E	4C	7E	6E	6F
4	7C	C1	C2	C3	C4	C5	C6	C7	C8	C9	D1	D2	D3	D4	D5	D6
5	D7	D8	D9	E2	E3	E4	E5	E6	E7	E8	E9	B1	B2	BB	BA	6D
6	79	81	82	83	84	85	86	87	88	89	91	92	93	94	95	96
7	97	98	99	A2	A3	A4	A5	A6	A7	A8	A9	C0	4F	D0	BC	07
8	68	DC	51	42	43	44	47	48	52	53	54	57	56	58	63	67
9	71	9C	9E	CB	CC	CD	DB	DD	DF	EC	FC	70	4A	80	3F	B4
A	45	55	CE	DE	49	69	9A	9B	AB	AF	5F	B8	B7	AA	8A	8B
B	3F	3F	3F	3F	3F	65	62	64	3F	3F	3F	3F	3F	B0	5B	3F
C	3F	3F	3F	3F	3F	3F	46	66	3F	3F	3F	3F	3F	3F	3F	9F
D	8C	AC	72	73	74	DA	75	76	77	3F	3F	3F	3F	6A	78	3F
E	EE	59	EB	ED	CF	EF	A0	8E	AE	FE	FB	FD	8D	AD	A1	BE
F	CA	8F	BF	B9	B6	B5	3F	9D	90	BD	B3	3F	FA	EA	3F	FF

EBCDIC to ASCII Translation for Japanese English

This chart is interpreted from left to right, and top to bottom. For example, if the first EBCDIC Digit is A, and the second EBCDIC Digit is F, you would find the A in the left column, and then locate the F in the top column. The point where the two columns meet would then be what the EBCDIC value would be translated to in ASCII representation. For example, the EBCDIC value AF would be translated to ASCII value A9.

Note: All digits are hexadecimal.

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0	00	01	02	03	1A	09	0A	7F	1A	1A	0A	0B	0C	0D	0E	0F
1	10	11	12	13	1A	0A	08	1A	18	19	1A	1A	1A	1A	1A	1A
2	1A	1A	1A	1A	1A	0A	17	1B	1A	1A	1A	1A	1A	05	06	07
3	1A	1A	16	1A	1A	1A	1A	04	1A	1A	1A	1A	14	15	1A	1A
4	20	1A	83	84	85	A0	C6	86	87	A4	9C	2E	3C	28	2B	7C
5	26	82	88	89	8A	A1	8C	8B	8D	E1	21	BE	2A	29	3B	AA
6	2D	2F	B6	8E	B7	B5	C7	8F	80	A5	DD	2C	25	5F	3E	3F
7	9B	90	D2	D3	D4	D6	D7	D8	DE	60	3A	23	40	27	3D	22
8	9D	61	62	63	64	65	66	67	68	69	AE	AF	D0	EC	E7	F1
9	F8	6A	6B	6C	6D	6E	6F	70	71	72	A6	A7	91	F7	92	CF
A	E6	EE	73	74	75	76	77	78	79	7A	AD	A8	D1	ED	E8	A9
B	BD	5B	5C	FA	9F	F5	F4	AC	AB	F3	5E	5D	7E	F9	EF	F2
C	7B	41	42	43	44	45	46	47	48	49	F0	93	94	95	A2	E4
D	7D	4A	4B	4C	4D	4E	4F	50	51	52	D5	96	81	97	A3	98
E	24	1A	53	54	55	56	57	58	59	5A	FD	E2	99	E3	E0	E5
F	30	31	32	33	34	35	36	37	38	39	FC	EA	9A	EB	E9	FF

ASCII to EBCDIC Translation for Portuguese

This chart is interpreted from left to right, and top to bottom. For example, if the first ASCII Digit is A, and the second ASCII Digit is F, you would find the A in the left column, and then locate the F in the top column. The point where the two columns meet would then be what the ASCII value would be translated to in EBCDIC representation. For example, the ASCII value AF would be translated to EBCDIC value 8B.

Note: All digits are hexadecimal.

	0	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**A**	**B**	**C**	**D**	**E**	**F**
0	00	01	02	03	37	2D	2E	2F	16	05	0A	0B	0C	0D	0E	0F
1	10	11	12	13	3C	3D	32	26	18	19	3F	27	3F	3F	3F	3F
2	40	4F	7F	66	5B	6C	50	7D	4D	5D	5C	4E	6B	60	4B	61
3	F0	F1	F2	F3	F4	F5	F6	F7	F8	F9	7A	5E	4C	7E	6E	6F
4	EF	C1	C2	C3	C4	C5	C6	C7	C8	C9	D1	D2	D3	D4	D5	D6
5	D7	D8	D9	E2	E3	E4	E5	E6	E7	E8	E9	4A	68	5A	5F	6D
6	79	81	82	83	84	85	86	87	88	89	91	92	93	94	95	96
7	97	98	99	A2	A3	A4	A5	A6	A7	A8	A9	46	BB	BE	48	07
8	E0	DC	51	42	43	44	47	A1	52	53	54	57	56	58	63	67
9	71	9C	9E	CB	CC	CD	DB	DD	DF	EC	FC	70	B1	80	3F	B4
A	45	55	CE	DE	49	69	9A	9B	AB	AF	BA	B8	B7	AA	8A	8B
B	3F	3F	3F	3F	3F	65	62	64	3F	3F	3F	3F	3F	B0	B2	3F
C	3F	3F	3F	3F	3F	3F	C0	7B	3F	3F	3F	3F	3F	3F	3F	9F
D	8C	AC	72	73	74	DA	75	76	77	3F	3F	3F	3F	CF	78	3F
E	EE	59	EB	ED	6A	7C	A0	8E	AE	FE	FB	FD	8D	AD	BC	D0
F	CA	8F	BF	B9	B6	B5	3F	9D	90	BD	B3	3F	FA	EA	3F	FF

EBCDIC to ASCII Translation for Portuguese

This chart is interpreted from left to right, and top to bottom. For example, if the first EBCDIC Digit is A, and the second EBCDIC Digit is F, you would find the A in the left column, and then locate the F in the top column. The point where the two columns meet would then be what the EBCDIC value would be translated to in ASCII representation. For example, the EBCDIC value AF would be translated to ASCII value A9.

Note: All digits are hexadecimal.

	0	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**A**	**B**	**C**	**D**	**E**	**F**
0	00	01	02	03	1A	09	0A	7F	1A	1A	0A	0B	0C	0D	0E	0F
1	10	11	12	13	1A	0A	08	1A	18	19	1A	1A	1A	1A	1A	1A
2	1A	1A	1A	1A	1A	0A	17	1B	1A	1A	1A	1A	1A	05	06	07
3	1A	1A	16	1A	1A	1A	1A	04	1A	1A	1A	1A	14	15	1A	1A
4	20	1A	83	84	85	A0	7B	86	7E	A4	5B	2E	3C	28	2B	21
5	26	82	88	89	8A	A1	8C	8B	8D	E1	5D	24	2A	29	3B	5E
6	2D	2F	B6	8E	B7	B5	23	8F	5C	A5	E4	2C	25	5F	3E	3F
7	9B	90	D2	D3	D4	D6	D7	D8	DE	60	3A	C7	E5	27	3D	22
8	9D	61	62	63	64	65	66	67	68	69	AE	AF	D0	EC	E7	F1
9	F8	6A	6B	6C	6D	6E	6F	70	71	72	A6	A7	91	F7	92	CF
A	E6	87	73	74	75	76	77	78	79	7A	AD	A8	D1	ED	E8	A9
B	BD	9C	BE	FA	9F	F5	F4	AC	AB	F3	AA	7C	EE	F9	7D	F2
C	C6	41	42	43	44	45	46	47	48	49	F0	93	94	95	A2	DD
D	EF	4A	4B	4C	4D	4E	4F	50	51	52	D5	96	81	97	A3	98
E	80	1A	53	54	55	56	57	58	59	5A	FD	E2	99	E3	E0	40
F	30	31	32	33	34	35	36	37	38	39	FC	EA	9A	EB	E9	FF

ASCII to EBCDIC Translation for Spanish–speaking

This chart is interpreted from left to right, and top to bottom. For example, if the first ASCII Digit is A, and the second ASCII Digit is F, you would find the A in the left column, and then locate the F in the top column. The point where the two columns meet would then be what the ASCII value would be translated to in EBCDIC representation. For example, the ASCII value AF would be translated to EBCDIC value 8B.

Note: All digits are hexadecimal.

	0	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**A**	**B**	**C**	**D**	**E**	**F**
0	00	01	02	03	37	2D	2E	2F	16	15	0A	0B	0C	0D	0E	0F
1	10	11	12	13	3C	3D	32	26	18	19	3F	27	3F	3F	3F	3F
2	40	BB	7F	69	5B	6C	50	7D	4D	5D	5C	4E	6B	60	4B	61
3	F0	F1	F2	F3	F4	F5	F6	F7	F8	F9	7A	5E	4C	7E	6E	6F
4	7C	C1	C2	C3	C4	C5	C6	C7	C8	C9	D1	D2	D3	D4	D5	D6
5	D7	D8	D9	E2	E3	E4	E5	E6	E7	E8	E9	4A	E0	5A	BA	6D
6	79	81	82	83	84	85	86	87	88	89	91	92	93	94	95	96
7	97	98	99	A2	A3	A4	A5	A6	A7	A8	A9	C0	4F	D0	BD	07
8	68	DC	51	42	43	44	47	48	52	53	54	57	56	58	63	67
9	71	9C	9E	CB	CC	CD	DB	DD	DF	EC	FC	70	B1	80	3F	B4
A	45	55	CE	DE	6A	7B	9A	9B	AB	AF	5F	B8	B7	AA	8A	8B
B	3F	3F	3F	3F	3F	65	62	64	3F	3F	3F	3F	3F	B0	B2	3F
C	3F	3F	3F	3F	3F	3F	46	66	3F	3F	3F	3F	3F	3F	3F	9F
D	8C	AC	72	73	74	DA	75	76	77	3F	3F	3F	3F	49	78	3F
E	EE	59	EB	ED	CF	EF	A0	8E	AE	FE	FB	FD	8D	AD	BC	BE
F	CA	8F	BF	B9	B6	B5	3F	9D	90	A1	B3	3F	FA	EA	3F	FF

EBCDIC to ASCII Translation for Spanish–speaking

This chart is interpreted from left to right, and top to bottom. For example, if the first EBCDIC Digit is A, and the second EBCDIC Digit is F, you would find the A in the left column, and then locate the F in the top column. The point where the two columns meet would then be what the EBCDIC value would be translated to in ASCII representation. For example, the EBCDIC value AF would be translated to ASCII value A9.

Note: All digits are hexadecimal.

	0	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**A**	**B**	**C**	**D**	**E**	**F**
0	00	01	02	03	1A	09	0A	7F	1A	1A	0A	0B	0C	0D	0E	0F
1	10	11	12	13	1A	0A	08	1A	18	19	1A	1A	1A	1A	1A	1A
2	1A	1A	1A	1A	1A	0A	17	1B	1A	1A	1A	1A	1A	05	06	07
3	1A	1A	16	1A	1A	1A	1A	04	1A	1A	1A	1A	14	15	1A	1A
4	20	1A	83	84	85	A0	C6	86	87	DD	5B	2E	3C	28	2B	7C
5	26	82	88	89	8A	A1	8C	8B	8D	E1	5D	24	2A	29	3B	AA
6	2D	2F	B6	8E	B7	B5	C7	8F	80	23	A4	2C	25	5F	3E	3F
7	9B	90	D2	D3	D4	D6	D7	D8	DE	60	3A	A5	40	27	3D	22
8	9D	61	62	63	64	65	66	67	68	69	AE	AF	D0	EC	E7	F1
9	F8	6A	6B	6C	6D	6E	6F	70	71	72	A6	A7	91	F7	92	CF
A	E6	F9	73	74	75	76	77	78	79	7A	AD	A8	D1	ED	E8	A9
B	BD	9C	BE	FA	9F	F5	F4	AC	AB	F3	5E	21	EE	7E	EF	F2
C	7B	41	42	43	44	45	46	47	48	49	F0	93	94	95	A2	E4
D	7D	4A	4B	4C	4D	4E	4F	50	51	52	D5	96	81	97	A3	98
E	5C	1A	53	54	55	56	57	58	59	5A	FD	E2	99	E3	E0	E5
F	30	31	32	33	34	35	36	37	38	39	FC	EA	9A	EB	E9	FF

ASCII to EBCDIC Translation for US English

This chart is interpreted from left to right, and top to bottom. For example, if the first ASCII Digit is A, and the second ASCII Digit is F, you would find the A in the left column, and then locate the F in the top column. The point where the two columns meet would then be what the ASCII value would be translated to in EBCDIC representation. For example, the ASCII value AF would be translated to EBCDIC value 57.

Note: All digits are hexadecimal.

	0	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**A**	**B**	**C**	**D**	**E**	**F**
0	00	01	02	03	37	2D	2E	2F	16	05	25	0B	0C	0D	0E	0F
1	10	11	12	13	3C	3D	32	26	18	19	3F	27	1C	1D	1E	1F
2	40	5A	7F	7B	5B	6C	50	7D	4D	5D	5C	4E	6B	60	4B	61
3	F0	F1	F2	F3	F4	F5	F6	F7	F8	F9	7A	5E	4C	7E	6E	6F
4	7C	C1	C2	C3	C4	C5	C6	C7	C8	C9	D1	D2	D3	D4	D5	D6
5	D7	D8	D9	E2	E3	E4	E5	E6	E7	E8	E9	4A	E0	4F	5F	6D
6	79	81	82	83	84	85	86	87	88	89	91	92	93	94	95	96
7	97	98	99	A2	A3	A4	A5	A6	A7	A8	A9	C0	6A	D0	A1	07
8	20	21	22	23	24	15	06	17	28	29	2A	2B	2C	09	0A	1B
9	30	31	1A	33	34	35	36	08	38	39	3A	3B	04	14	3E	E1
A	41	42	43	44	45	46	47	48	49	51	52	53	54	55	56	57
B	58	59	62	63	64	65	66	67	68	69	70	71	72	73	74	75
C	76	77	78	80	8A	8B	8C	8D	8E	8F	90	9A	9B	9C	9D	9E
D	9F	A0	AA	AB	AC	AD	AE	AF	BO	B1	B2	B3	B4	B5	B6	B7
E	B8	B9	BA	BB	BC	BD	BE	BF	CA	CB	CC	CD	CE	CF	DA	DB
F	DC	DD	DE	DF	EA	EB	EC	ED	EE	EF	FA	FB	FC	FD	FE	FF

EBCDIC to ASCII Translation for US English

This chart is interpreted from left to right, and top to bottom. For example, if the first EBCDIC Digit is A, and the second EBCDIC Digit is F, you would find the A in the left column, and then locate the F in the top column. The point where the two columns meet would then be what the EBCDIC value would be translated to in ASCII representation. For example, the EBCDIC value AF would be translated to ASCII value D7.

Note: All digits are hexadecimal.

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0	00	01	02	03	9C	09	86	7F	97	8D	8E	0B	0C	0D	0E	0F
1	10	11	12	13	9D	85	08	87	18	19	92	8F	1C	1D	1E	1F
2	80	81	82	83	84	0A	17	1B	88	89	8A	8B	8C	05	06	07
3	90	91	16	93	94	95	96	04	98	99	9A	9B	14	15	9E	1A
4	20	A0	A1	A2	A3	A4	A5	A6	A7	A8	5B	2E	3C	28	2B	5D
5	26	A9	AA	AB	AC	AD	AE	AF	B0	B1	21	24	2A	29	3B	5E
6	2D	2F	B2	B3	B4	B5	B6	B7	B8	B9	7C	2C	25	5F	3E	3F
7	BA	BB	BC	BD	BE	BF	C0	C1	C2	60	3A	23	40	27	3D	22
8	C3	61	62	63	64	65	66	67	68	69	C4	C5	C6	C7	C8	C9
9	CA	6A	6B	6C	6D	6E	6F	70	71	72	CB	CC	CD	CE	CF	D0
A	D1	7E	73	74	75	76	77	78	79	7A	D2	D3	D4	D5	D6	D7
B	D8	D9	DA	DB	DC	DD	DE	DF	E0	E1	E2	E3	E4	E5	E6	E7
C	7B	41	42	43	44	45	46	47	48	49	E8	E9	EA	EB	EC	ED
D	7D	4A	4B	4C	4D	4E	4F	50	51	52	EE	EF	F0	F1	F2	F3
E	5C	9F	53	54	55	56	57	58	59	5A	F4	F5	F6	F7	F8	F9
F	30	31	32	33	34	35	36	37	38	39	FA	FB	FC	FD	FE	FF

ASCII to EBCDIC Translation for UK English

This chart is interpreted from left to right, and top to bottom. For example, if the first ASCII Digit is A, and the second ASCII Digit is F, you would find the A in the left column, and then locate the F in the top column. The point where the two columns meet would then be what the ASCII value would be translated to in EBCDIC representation. For example, the ASCII value AF would be translated to ASCII value 8B.

Note: All digits are hexadecimal.

	0	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**A**	**B**	**C**	**D**	**E**	**F**
0	00	01	02	03	37	2D	2E	2F	16	05	0A	0B	0C	0D	0E	0F
1	10	11	12	13	3C	3D	32	26	18	19	3F	27	3F	3F	3F	3F
2	40	5A	7F	7B	4A	6C	50	7D	4D	5D	5C	4E	6B	60	4B	61
3	F0	F1	F2	F3	F4	F5	F6	F7	F8	F9	7A	5E	4C	7E	6E	6F
4	7C	C1	C2	C3	C4	C5	C6	C7	C8	C9	D1	D2	D3	D4	D5	D6
5	D7	D8	D9	E2	E3	E4	E5	E6	E7	E8	E9	B1	E0	BB	BA	6D
6	79	81	82	83	84	85	86	87	88	89	91	92	93	94	95	96
7	97	98	99	A2	A3	A4	A5	A6	A7	A8	A9	C0	4F	D0	BC	07
8	68	DC	51	42	43	44	47	48	52	53	54	57	56	58	63	67
9	71	9C	9E	CB	CC	CD	DB	DD	DF	EC	FC	70	5B	80	3F	B4
A	45	55	CE	DE	49	69	9A	9B	AB	AF	5F	B8	B7	AA	8A	8B
B	3F	3F	3F	3F	3F	65	62	64	3F	3F	3F	3F	3F	B0	B2	3F
C	3F	3F	3F	3F	3F	3F	46	66	3F	3F	3F	3F	3F	3F	3F	9F
D	8C	AC	72	73	74	DA	75	76	3F	3F	3F	3F	3F	6A	78	3F
E	EE	59	EB	ED	CF	EF	A0	8E	AE	FE	FB	FD	8D	AD	A1	BE
F	CA	8F	BF	B9	B6	B5	3F	9D	90	BD	B3	3F	FA	EA	3F	FF

EBCDIC to ASCII Translation for UK English

This chart is interpreted from left to right, and top to bottom. For example, if the first EBCDIC Digit is A, and the second EBCDIC Digit is F, you would find the A in the left column, and then locate the F in the top column. The point where the two columns meet would then be what the/ EBCDIC value would be translated to in ASCII representation. For example, the EBCDIC value AF would be translated to ASCII value A9.

Note: All digits are hexadecimal.

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0	00	01	02	03	1A	09	0A	7F	1A	1A	0A	0B	0C	0D	0E	0F
1	10	11	12	13	1A	0A	08	1A	18	19	1A	1A	1A	1A	1A	1A
2	1A	1A	1A	1A	1A	0A	17	1B	1A	1A	1A	1A	1A	05	06	07
3	1A	1A	16	1A	1A	1A	1A	04	1A	1A	1A	1A	14	15	1A	1A
4	20	1A	83	84	85	A0	C6	86	87	A4	24	2E	3C	28	2B	7C
5	26	82	88	89	8A	A1	8C	8B	8D	E1	21	9C	2A	29	3B	AA
6	2D	2F	B6	8E	B7	B5	C7	8F	80	A5	DD	2C	25	5F	3E	3F
7	9B	90	D2	D3	D4	D6	D7	D8	DE	60	3A	23	40	27	3D	22
8	9D	61	62	63	64	65	66	67	68	69	AE	AF	D0	EC	E7	F1
9	F8	6A	6B	6C	6D	6E	6F	70	71	72	A6	A7	91	F7	92	CF
A	E6	EE	73	74	75	76	77	78	79	7A	AD	A8	D1	ED	E8	A9
B	BD	5B	BE	FA	9F	F5	F4	AC	AB	F3	5E	5D	7E	F9	EF	F2
C	7B	41	42	43	44	45	46	47	48	49	F0	93	94	95	A2	E4
D	7D	4A	4B	4C	4D	4E	4F	50	51	52	D5	96	81	97	A3	98
E	5C	1A	53	54	55	56	57	58	59	5A	FD	E2	99	E3	E0	E5
F	30	31	32	33	34	35	36	37	38	39	FC	EA	9A	EB	E9	FF

ASCII to EBCDIC Translation for US English (TEXT)

If you requested the TEXT option with the **emsend** command, this table is used for ASCII to EBCDIC translation.

This chart is interpreted from left to right, and top to bottom. For example, if the first ASCII Digit is A, and the second ASCII Digit is F, you would find the A in the left column, and then locate the F in the top column. The point where the two columns meet would then be what the ASCII value would be translated to in EBCDIC representation. For example, the ASCII value AF would be translated to EBCDIC value 3F.

Note: All digits are hexadecimal.

	0	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**A**	**B**	**C**	**D**	**E**	**F**
0	00	01	02	03	37	2D	2E	2F	16	05	0A	0B	0C	0D	0E	0F
1	10	11	12	13	3C	3D	32	26	18	19	3F	27	3F	3F	3F	3F
2	40	5A	7F	7B	5B	6C	50	7D	4D	5D	5C	4E	6B	60	4B	61
3	F0	F1	F2	F3	F4	F5	F6	F7	F8	F9	7A	5E	4C	7E	6E	6F
4	7C	C1	C2	C3	C4	C5	C6	C7	C8	C9	D1	D2	D3	D4	D5	D6
5	D7	D8	D9	E2	E3	E4	E5	E6	E7	E8	E9	AD	E0	BD	5F	6D
6	79	81	82	83	84	85	86	87	88	89	91	92	93	94	95	96
7	97	98	99	A2	A3	A4	A5	A6	A7	A8	A9	C0	4F	D0	A1	07
8	FD	73	69	8A	75	57	EB	56	8B	76	58	77	8C	59	B7	FC
9	CC	FA	9C	8D	78	62	8E	63	66	BA	BB	3F	43	3F	3F	3F
A	8F	9A	9B	3F	9D	DC	41	3F	3F	3F	52	3F	3F	3F	3F	3F
B	3F	3F	3F	29	3F	3F	3F	3F	3F	3F	3F	3F	3F	4A	44	3F
C	3F	3F	3F	3F	3F	3F	3F	3F	3F	3F	3F	3F	3F	3F	3F	3F
D	3F	3F	3F	3F	3F	3F	3F	3F	3F	3F	3F	3F	3F	3F	3F	3F
E	3F	3F	3F	3F	3F	3F	3F	3F	3F	3F	3F	3F	3F	3F	3F	55
F	3F	3F	3F	3F	3F	48	3F	3F	3F	3F	3F	3F	3F	3F	3F	FF

EBCDIC to ASCII Translation for US English (TEXT)

If you requested the TEXT option with the **emrcv** command, this table is used for EBCDIC to ASCII translation.

This chart is interpreted from left to right, and top to bottom. For example, if the first EBCDIC Digit is A, and the second EBCDIC Digit is F, you would find the A in the left column, and then locate the F in the top column. The point where the two columns meet would then be what the EBCDIC value would be translated to in ASCII representation. For example, the EBCDIC value AF would be translated to ASCII value 4A.

Note: All digits are hexadecimal.

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0	00	01	02	03	1A	09	0A	7F	1A	1A	0A	0B	0C	0D	0E	0F
1	10	11	12	13	1A	0A	08	1A	18	19	1A	1A	1A	1A	1A	1A
2	1A	1A	1A	1A	1A	0A	17	1B	1A	1A	1A	1A	1A	05	06	07
3	1A	1A	16	1A	1A	1A	1A	04	1A	1A	1A	1A	14	15	1A	1A
4	20	A6	E1	9C	BE	1A	1A	1A	F5	1A	BD	2E	3C	28	2B	7C
5	26	1A	AA	1A	22	EF	87	85	8A	8D	21	24	2A	29	3B	5E
6	2D	2F	95	97	61	6F	98	85	8A	82	BA	2C	25	5F	3E	3F
7	8D	95	97	81	87	84	89	8B	94	60	3A	23	40	27	3D	22
8	81	61	62	63	64	65	66	67	68	69	83	88	8C	93	96	A0
9	82	6A	6B	6C	6D	6E	6F	70	71	72	A1	A2	92	A4	41	45
A	49	7E	73	74	75	76	77	78	79	7A	4F	55	41	5B	59	4A
B	45	45	49	4F	55	59	53	8E	45	49	99	9A	41	5D	49	4F
C	7B	41	42	43	44	45	46	47	48	49	55	41	90	49	1A	1A
D	7D	4A	4B	4C	4D	4E	4F	50	51	52	4F	55	A5	1A	1A	1A
E	5C	1A	53	54	55	56	57	58	59	5A	1A	86	87	1A	1A	1A
F	30	31	32	33	34	35	36	37	38	39	91	1A	8F	80	1A	FF

Unsupported Keys

Unsupported Keys for Austrian/German

°	Degree Symbol	{	Left Brace
[	Left Bracket	}	Right Brace
]	Right Bracket	\	Reverse Slash
´	Acute Accent	@	At Sign
~	Tilde Accent	|	Pipe Symbol
µ	Mu Small	²	Superscript 2
³	Superscript 3	×	Multiplication Sign
÷	Division Sign		

Unsupported Keys for Belgian

|	Pipe Symbol	@	At Sign
²	Superscript 2	³	Superscript 3
§	Section Symbol	{	Left Brace
}	Right Brace	°	Degree Symbol
´	Acute Accent	µ	Greek Mu Small
£	Pounds Sterling	\	Reverse Slash
~	Tilde Accent		

Unsupported Keys for Canadian–French

|	OR Sign	\	Reverse Slash
±	Plus or Minus Sign	£	Pounds Sterling
¢	Cent Sign	¤	International Currency Symbol (ICS)
¬	NOT Symbol	|	Pipe Symbol
²	Superscript 2	³	Superscript 3
1/4	One Fourth	1/2	One Half
3/4	Three Fourths	§	Section Symbol
¶	Paragraph Symbol	[	Left Bracket

]	Right Bracket	~	Tilde Accent
{	Left Brace	}	Right Brace
«	Left Angle Quotes	»	Right Angle Quotes
°	Degree Symbol	µ	Micro Symbol
¯	Overline, Overbar	‗	Double Underscore

Additional Keys Available on the Canadian–French Keyboard

The following characters are not engraved on a key. They are composed of a non–spacing (dead key) character plus another character.

à À è È ù Ù	Characters with a grave accent
é É	Characters with an acute accent
â Â ê Ê î Î ô Ô û Û	Characters with a circumflex accent
ë Ë ï Ï ü Ü	Characters with a diaeresis accent
ç Ç	Characters with a cedilla accent

Unsupported Keys for Danish

1/2	One Half Fraction	§	Section Symbol
@	At SIgn	£	Pounds Sterling
$	Dollar Sign	{	Left Brace
[	Left Bracket	]	Right Bracket
}	Right Brace	´	Acute Accent
|	Pipe Symbol	¨	Diaeresis Accent
~	Tilde Accent		

Additional Key Available on the Danish Keyboard

The following character is not engraved on a key. It is composed of a non–spacing (dead key) character plus another character.

ü	Lowercase u with diaeresis accent

Unsupported Keys for UK English

^	Circumflex Accent	[	Left Bracket
]	Right Bracket	~	Tilde Accent

Additional Keys Available on the UK English Keyboard

The following character is not engraved on a key. It is composed of a non–spacing (dead key) character plus another character.

¯	Overbar (key 42_s)

Unsupported Keys for Finnish and Swedish

£	Pounds Sterling	$	Dollar Sign
`	Grave Accent	´	Acute Accent
×	Multiplication Sign	1/2	One Half Fraction
¦	Pipe Symbol	@	At Sign
#	Number Sign	[	Left Bracket
]	Right Bracket	{	Left Brace
}	Right Brace	~	Tilde Accent
¨	Diaeresis Accent	\	Reverse Slash
÷	Division Sign		

Unsupported Keys for French (AZERTY)

#	Number SIgn	{	Left Brace
[	Left Bracket	¦	Pipe Symbol
\	Reverse Slash	@	At Sign
]	Right Bracket	}	Right Brace
µ	Greek Mu Small	¤	International Currency Symbol (ICS)
~	Tilde Accent	²	Superscript 2

Unsupported Keys for Italian

\	Reverse Slash	¦	Pipe Symbol
{	Left Brace	}	Right Brace
~	Tilde Accent		Grave Accent
[	Left Bracket	]	Right Bracket
@	At Sign	#	Number Sign

Unsupported Keys for Japanese English

^	Circumflex Accent	^	Tilde Accent
[	Left Bracket	]	Right Bracket
\	Reverse Slash		

Additional Keys Available on the Japanese English Keyboard

The following characters are not engraved on a key. They are composed of a non–spacing (dead key) character plus another character.

|	Vertical Bar (key k1_c)
¦	Pipe Symbol (key k1_s)

¥	Yen (key k5_a)
£	Pounds Sterling (key k6_a)
¬	NOT Symbol (key k11_a)

Unsupported Keys for Norwegian

|	Pipe Symbol	§	Section Symbol
@	At Sign	£	Pounds Sterling
$	Dollar Sign	{	Left Brace
[	Left Bracket	}	Right Brace
]	Right Bracket	´	Acute Accent
¨	Diaeresis Accent	~	Tilde Accent

Additional Key Available on the Norwegian Keyboard

The following character is not engraved on a key. It is composed of a non–spacing (dead key) character plus another character.

ü	Lowercase u with diaeresis accent

Unsupported Keys for Portuguese

\	Reverse Slash	|	Pipe Symbol
@	At Sign	#	Number SIgn
£	Pounds Sterling	§	Section Symbol
{	Left Brace	}	Right Brace
«	Left Angle Quotes	»	Right Angle Quotes
¨	Diaeresis Accent	º	Ordinal Ind., Masculine
ª	Ordinal Ind., Feminine	~	Tilde Accent

Additional Keys Available on the Portuguese Keyboard

The following characters are not engraved on a key. They are composed of a non–spacing (dead key) character plus another character.

ã Ã õ Õ	Characters with a tilde accent

Unsupported Keys for Spanish–speaking Keyboard

´	Acute Accent	ç	c Cedilla, Small
Ç	c Cedilla, Capital	º	Ordinal Ind., Masculine
ª	Ordinal Ind., Feminine	!	Exclamation Point
•	Product Dot	#	Number Sign
~	Tilde Accent	¡	Inverted !
¿	Inverted ?	^	Circumflex Accent

Unsupported Keys for Swiss–French and Swiss–German

´	Acute Accent	§	Section Symbol
°	Degree Symbol	|	OR Symbol
@	At Sign	#	Number Sign
¬	NOT Sign	¦	Pipe Symbol
¢	Cent Sign	~	Tilde Accent
[	Left Bracket	]	Right Bracket
{	Left Brace	}	Right Brace
\	Reverse Slash		

Additional Keys Available on the Swiss–French and Swiss–German Keyboards

The following characters are not engraved on a key. They are composed of a non–spacing (dead key) character plus another character.

à è ì ò ù	Characters with a grave accent
â ê î ô û	Characters with a circumflex accent
ä Ä ë ï ö Ö ü Ü	Characters with a diaeresis accent

Chapter 4. AIX 3270 Host Connection Program/6000

The AIX 3270 Host Connection Program/6000 Licensed Program (HCON) allows your system to emulate a terminal or printer attached to an IBM System/370 system. This chapter discusses using and managing HCON. Information about using HCON includes definitions of emulator functions and special keys, and information about transferring files with HCON. Information about managing HCON includes definitions of important concepts, installation and configuration information, and information to help you customize and maintain HCON after it is set up on your system. Information about programming HCON is included in *AIX Communications Programming Concepts for IBM RISC System/6000.*

HCON Overview

The AIX 3270 Host Connection Program/6000 (HCON) emulates a 3278/79 display or a 3286/87 printer attached to an IBM System/370 computer. HCON operates in *sessions.* To use HCON to emulate a display, run a *display session.* To use HCON to emulate a printer, run a *printer session.*

At the beginning of a display session, the emulator acts as if you had just turned on the 3278/79 terminal. After you log in to the System/370 host, you can run commands and programs. For example, you might check for mail on the host. You can also use the **fxfer** command to upload and download files to and from the host as well as change the format of the data in the files as you transfer them. For example, you can translate files from ASCII to EBCDIC or add or remove carriage-return characters.

Using a printer session, you can print files from the host on a local printer as if it were attached directly to the host or save the files from the host in printable form in an AIX file to print or edit later.

HCON has special users. A user with root authority must register you as an HCON user before you can use HCON commands. After you have been registered, you must create at least one *session profile* before you can run HCON. (The person who registered you may also have created a default profile for you.)

The session profile specifies parameters for HCON emulator sessions. For example, the profile tells HCON the type of host you are connecting to, whether you are emulating a display or printer, the type of connection you are using to communicate with the host, the local printer name for a printer emulation, and so on. You can have up to 26 session profiles defined for yourself. The profiles allow you to connect to more than one host, to connect to the same host using different parameters, and to run simultaneous sessions on one or more hosts, with similar or differing parameters. Other HCON users can define up to 26 of their own profiles, with different settings.

Most of the things that you can change about the way HCON works are in the session profile. However, you can also set up *color* and *keyboard* definition tables. The HCON color definition utility allows you to change the way host field and character attributes are displayed on your terminal. The keyboard definition utility allows you to change the keys you use at your terminal to communicate with the host. As with sessions, you can have multiple color and keyboard tables. Simply specify which tables to use in the session profiles you create.

The HCON Application Programming Interface allows programmers to write applications that communicate with a System/370 host.

Related Information

Understanding HCON Sessions on page 4–20 describes HCON sessions and session profiles.

Run the **smit hcon** command to view, create, or modify HCON session profiles.

How to Customize the HCON Color Definition Table on page 4–59 and How to Customize the HCON Keyboard Definition Table on page 4–60 explain how to change HCON colors and keys.

The **e789** command starts HCON emulation sessions. The **fxfer** command performs HCON file transfers.

HCON Introduction for System Management on page 4–18 discusses HCON for system management personnel. HCON Overview for Programming in *Communications Programming Concepts* discusses HCON for programmers.

HCON Emulator Overview

The AIX 3270 Host Connection Program/6000 (HCON) **e789** Command emulates a 3278/79 display or a 3286/87 printer attached to an IBM System/370 computer. An emulator is an application that allows your machine to imitate another device. Emulators provide the functions of the device being emulated as if you were actually using that device.

At the beginning of a display session, the emulator acts as if you had just turned on the 3278/79 terminal. After you log in to the System/370 host, you can run commands and programs. For example, you might check for mail on the host. You can also upload and download files to and from the host as well as change the format of the data in the files as you transfer them. For example, you can translate files from ASCII to EBCDIC or add or remove carriage-return characters.

Using a printer session, you can print files from the host on a local printer as if it were attached directly to the host or save the files from the host in printable form in an AIX file to print or edit later.

The following information is provided to assist you in using the HCON **e789** Command:

- Locking and Unlocking the Keyboard for **e789** on page 4–3
- **e789** Special Keys on page 4–3
- Using Default e789 Keyboards on page 4–6
  - Using **e789** on RISC System/6000 Keyboards on page 4–6
  - Using **e789** on IBM 3151, IBM 3161, IBM 3162, and IBM 3163 Keyboards on page 4–8
  - Using **e789** on DEC VT100 Keyboards on page 4–9
  - Using **e789** on DEC VT220 Keyboards on page 4–10
  - Using **e789** on WYSE WY–50 Keyboards on page 4–11
- Understanding the HCON Operator Information Area (OIA) on page 4–13.

Locking and Unlocking the Keyboard for e789

IBM AIX 3270 Host Connection Program/6000 has additional features to prevent the keyboard from locking and to help unlock the keyboard. When an attribute is reached while typing data into a field, the cursor is automatically moved to the next unprotected field location instead of locking the keyboard and leaving the cursor on the attribute.

Cursor movement keys unlock the keyboard in addition to performing their normal functions. Most of the time, when the keyboard is locked, it is because an attempt was made to enter data at the wrong place. Thus, any key that moves the cursor also unlocks the keyboard. The **RESET** key also unlocks the keyboard. The following cursor movement keys can be used to unlock the keyboard:

Cursor Left
Cursor Right
Cursor Down
Cursor Up
Cursor Home
End Field
Left End Field Key
Forward Tab
Backward Tab
Newline
Erase Input
Reset
Backspace
Forward Word Tab
Backward Word Tab

In addition, if the keyboard is locked because the MORE THAN condition (X 0 >) in the Operator Information Area (OIA) is displayed as the result of trying to insert characters into a full field, the **INSERT** key, the **DELETE** key, and the **DELWORD** key can be used to unlock the keyboard (as well as performing their normal functions).

e789 Special Keys

The following is a description of special **e789** keys. These keys include functions that appear on the 3278/79 keyboards, such as **PF1** and **ENTER**, and special emulator keys, such as **DELWORD** and **FTABW**, that move the cursor, add and delete text, and switch active emulator screens.

Each special emulator key has a default key sequence. By using the **e789kdef** keyboard redefinition utility, different key sequences may be assigned to a specific key function.

Attention Keys
Include the **CLEAR**, **ENTER**, **PF1** to **PF24**, **PA1** to **PA3**, and **SYSREQ** keys.

BACKSPACE and **BACKF**
The IBM AIX 3270 Host Connection Program/6000 has two types of destructive backspace keys: **BACKSPACE** and **BACKF**. Both delete the character under the cursor and move the remaining characters in the field to the left. If a field contains several lines, characters are moved from line to another. The difference in function between the **BACKSPACE** and the **BACKF** keys is the processing performed when an attribute byte is deleted.

The **BACKSPACE** key moves the cursor to the left when the cursor is on a protected screen location. It does not lock the keyboard or jump fields. When the **BACKF** key detects an attribute byte while backspacing, it moves the cursor to the last non–null, non–space character of the previous unprotected field and erases it.

CHGDIS Switches you to another display if two displays are operating. If the auxiliary display is not turned on or if the secondary display card has no display attached to it, the **CHGDIS** key appears to lock up the current physical display. This happens because the **e789** process is trying to use the secondary display and is not aware of its unavailability.

Cursor Movement Keys

These keys are:

LEFT	Moves the cursor to the left one position.
LLEFT	Moves the cursor to the left two positions.
RIGHT	Moves the cursor to the right one position.
RRIGHT	Moves the cursor to the right two positions.
DOWN	Moves the cursor down one position.
DDOWN	Moves the cursor down two positions.
UP	Moves the cursor up one position.
UUP	Moves the cursor up two positions.

DELETE Deletes the character at the cursor. This key will also unlock the keyboard if the MORE THAN (X 0 > in OIA) condition is displayed as a result of trying to insert too much data into a field.

DELWORD Deletes characters from the cursor to the end of the word. The words following are shifted to the left. This deletion includes punctuation and one space or null (if there is one) following the word. For a partial word deletion, the trailing space or null is not deleted. The cursor is left at its current location. For word deletion, a word or field does not "wrap" the screen. If the cursor is on a null or a space, the null or space is deleted. If the cursor is on an attribute or a protected field, input is inhibited and the Go Elsewhere symbol (X < 0 >) is displayed in the Operator Information Area (OIA).

DOIA Toggles the OIA on or off on the ASCII terminals. It performs no function when the **e789** command is executed on displays with a screen that has more than 24 rows.

DUP Informs the application program that a duplicate operation is indicated for the rest of the field in which it is located. **DUP** is displayed as a blank and tabs forward to the next field. If the cursor is on an attribute or a protected field, input is inhibited and the Go Elsewhere Symbol (X < 0 >) is displayed in the Operator Information Area (OIA).

ENDF Moves the cursor to one position past the last non–null, non–space character in a field. If a field is full, the cursor is moved to the attribute byte of the next field.

EOF Clears the character location occupied by the cursor and all remaining character locations in that field to nulls if the cursor is located in an unprotected field.

ERASE Clears all unprotected character locations to nulls, sets all character attributes to their default values, and repositions the cursor to the first unprotected character location on the display.

FMARK Informs the application program of the end of a field in an unformatted buffer or subfield of a formatted buffer. **FMARK** is displayed as a blank. If the cursor is positioned at a field attribute location or is within a protected field, input is inhibited and the Go Elsewhere Symbol (X < 0 >) is displayed in the Operator Information Area (OIA).

FTABW and **BTABW**
Word tab keys allow the cursor to move to the next or previous word with a single keystroke. A word is one or more non–null, non–space characters. Words are separated by nulls or spaces. This kind of tabbing places the cursor on the first character of the next or previous word or at the upper leftmost position of the display, if no words exist. This tabbing ignores field attributes and only looks at the characters displayed on the screen. Thus, it is possible to tab by word through protected field locations. Normal 3270 field tab keys are provided for moving the cursor to the next and previous unprotected fields.

HOME Positions the cursor at the first unprotected character location on the display.

INSERT Operates as it does on a 3278/79, with several additions. The **INSERT** key puts the terminal into insert mode. This is indicated by a carat (^) in the Operator Information Area (at the bottom of the screen). All subsequent keyed characters are inserted rather than overstruck. If the field becomes full while you are in Insert mode and there are trailing blanks at the end of the field, those blanks are shifted out of the field. Characters can then be entered until there are no nulls or trailing blanks left in the field. Insert mode may be terminated by pressing the **RESET** key. The **INSERT** key also acts as a toggle, so that pressing this key while in insert mode resets the terminal to replacement mode.

JOBCTRL Allows the user to suspend the emulator if run from a shell which supports job control. When the **e789** process is suspended, the keyboard and display are released to the parent shell, and the host may modify the internal presentation space of the emulation session.

The **e789** process cannot be put into the background. If the user attempts to do this, the **e789** process will remain suspended and a message will be displayed. To put the **e789** process back into the foreground, the user must type:

```
fg <job #>
```

LENDF Moves the cursor to the first non–null, non–space character in a field. If a field is empty, the cursor is moved to the first position of the current field.

NEWLINE Moves the cursor to the next line and wraps to the top of the screen for the last line.

NEXT Makes the next session the active session. For example, if four sessions are started by entering `e789 abcd`, four sessions with session names of `a`, `b`, `c`, and `d` are established. Session `a` is the active session and is usable on the display. Pressing the **NEXT** hot key causes session `b` to become active. Pressing the **NEXT** key three more times activates the `c`, `d`, and `a` sessions in succession. This key performs no function if only one session is available.

POREST Emulates power–on reset. This is equivalent to turning on the power on an IBM 3278/79 display. The **ENTER** key may need to be pressed before the logo is redisplayed.

PRINT Prints the contents of the display on the printer that is associated with the IBM RISC System/6000.

QUIT Terminates the emulator session, the same as turning off an IBM 3278/79 display.

REFRESH Refreshes the current screen on the active display session.

REPLS Replaces the contents of the replace file specified in the session profile with the contents of the display.

RESET Unlocks the keyboard after an attempt to type data in the wrong place on the screen (for example, in a protected screen location) or an attempt to insert too much data into a field. In addition to the **RESET** key, any of the Cursor Movement Keys may be used to reset the keyboard.

SAVES Appends the contents of the display to the save file specified in the session profile.

SHELL Invokes an AIX subshell specified by the SHELL environment variable. While the subshell is executing, the host may modify the internal presentation space of the emulation session. When the subshell is terminated, the emulation session regains use of the virtual terminal.

SPACEOF Performs the same functions as the 3270 **EOF** key except that the space character performs the erasure, rather than the null character.

TAB and **BTAB**
These are the normal 3270 field tab keys for moving the cursor to the next and previous unprotected fields.

Using Default e789 Keyboards

The IBM AIX 3270 Host Connection Program/6000 is shipped with default keyboard mappings that allow you to execute the **e789** command from an HFT and supported ASCII terminals. Commonly used functions are in the same position on all keyboard types. You can redefine the IBM AIX 3270 Host Connection Program/6000 keyboard by using the **e789kdef** command. The following sections describe the keyboards which are supported by the IBM AIX 3270 Host Connection Program/6000. Only special **e789** function keys are defined in the tables.

Functions are defined for five groups of keyboards, as follows:

- Using **e789** on RISC System/6000 Keyboards on page 4–6
- Using **e789** on IBM 3151, IBM 3161, IBM 3162, and IBM 3163 Keyboards on page 4–8
- Using **e789** on DEC VT100 Keyboards on page 4–9
- Using **e789** on DEC VT220 Keyboards on page 4–10
- Using **e789** on WYSE WY–50 Keyboards on page 4–11.

Using e789 on RISC System/6000 Keyboards

The following chart shows the default IBM AIX 3270 Host Connection Program/6000 key functions on RISC System/6000 console keyboards. The position of the keys on the console keyboard does not change when executing the emulator locally, through a window in IBM AIXwindows, or remotely through a TCP/IP connection. See e789 Special Keys on page 4–3 for a description of the functions.

e789 Key Functions on RISC System/6000 Keyboards			
Function	**Press**	**Function**	**Press**
TAB	Tab	PF13	Shift – F1
BTAB	Shift – Tab	PF14	Shift – F2
ENTER	Ctrl/Act	PF15	Shift – F3
SYSREQ	Alt – Print Screen	PF16	Shift – F4
CLEAR	Pause	PF17	Shift – F5
BACKSPACE	Backspace	PF18	Shift – F6
CHGDIS	Ctrl – keypad 6	PF19	Shift – F7
INSERT	Insert	PF20	Shift – F8
DELETE	Delete	PF21	Shift – F9
HOME	Home	PF22	Shift – F10
PA1	Page Up	PF23	Shift – F11
PA2	Page Down	PF24	Shift – F12
PA3	Ctrl – Right Arrow	PRINT	Print Screen
SPACEOF	End	SAVES	Shift – Scroll Lock
DOWN	Down Arrow	REPLS	Scroll Lock
DDOWN	Shift – Down Arrow	BTABW	Ctrl – A
RIGHT	Right Arrow	EOF	Ctrl – B
RRIGHT	Shift – Right Arrow	SHELL	Ctrl – C
LEFT	Left Arrow	QUIT	Ctrl – D
LLEFT	Shift – Left Arrow	ERASE	Ctrl – E
UP	Up Arrow	FTABW	Ctrl – F
UUP	Shift – Up Arrow	ENDF	Ctrl – G
PF1	F1	BACKF	Ctrl – K
PF2	F2	LENDF	Ctrl – L
PF3	F3	NEWLINE	Ctrl – M
PF4	F4	FMARK	Ctrl – N
PF5	F5	NEXT	Ctrl – O
PF6	F6	POREST	Ctrl – P
PF7	F7	RESET	Ctrl – R
PF8	F8	REFRESH	Ctrl – T
PF9	F9	DUP	Ctrl – U
PF10	F10	DELWORD	Ctrl – W
PF11	F11	DOIA	Ctrl – X
PF12	F12	JOBCTRL	Ctrl – Z
		NEWLINE	Enter

Using e789 on IBM 3151, IBM 3161, IBM 3162, and IBM 3163 Keyboards

The following chart shows the default IBM AIX 3270 Host Connection Program/6000 key functions on IBM 3151, IBM 3161, IBM 3162, and IBM 3163 terminal keyboards. See e789 Special Keys on page 4–3 for a description of the functions.

The IBM 3151, IBM 3161, IBM 3162, and IBM 3163 terminals must be configured as follows for the **e789** process to function properly:

- The turnaround character must be 0x0a
- SEND must be set to PAGE
- INSERT must be set to SPACE

e789 Key Functions on IBM 3151, 3161, 3162, and 3163 Keyboards
Part 1 of 2

Function	Press	Function	Press
TAB	Tab	PF13	Shift – F1
BTAB	Shift – Tab	PF14	Shift – F2
ENTER	Send	PF15	Shift – F3
SYSREQ	EraseEOF	PF16	Shift – F4
CLEAR	Clear	PF17	Shift – F5
BACKSPACE	Backspace	PF18	Shift – F6
CHGDIS	N/A	PF19	Shift – F7
INSERT	Insert	PF20	Shift – F8
DELETE	Delete	PF21	Shift – F9
HOME	Home	PF22	Shift – F10
PA1	PA1	PF23	Shift – F11
PA2	PA2	PF24	Shift – F12
PA3	PA3	PRINT	Send Line
SPACEOF	Ctrl – Clear	SAVES	Ctrl – Insert
DOWN	Down Arrow	REPLS	Ctrl – Delete
DDOWN	N/A	BTABW	Ctrl – A
RIGHT	Right Arrow	EOF	Ctrl – B
RRIGHT	N/A	SHELL	Ctrl – C
LEFT	Left Arrow	QUIT	Ctrl – D
LLEFT	N/A	ERASE	Ctrl – E
UP	Up Arrow	FTABW	Ctrl – F
UUP	N/A	ENDF	Ctrl – G
NEWLINE	Enter	BACKF	Ctrl – K
PF1	F1	LENDF	Ctrl – L
PF2	F2	NEWLINE	Ctrl – M
PF3	F3	FMARK	Ctrl – N
PF4	F4	NEXT	Ctrl – O
PF5	F5	POREST	Ctrl – P

e789 Key Functions on IBM 3151, 3161, 3162, and 3163 Keyboards Part 2 of 2			
Function	**Press**	**Function**	**Press**
PF6	F6	RESET	Ctrl – R
PF7	F7	REFRESH	Ctrl – T
PF8	F8	DUP	Ctrl – U
PF9	F9	DELWORD	Ctrl – W
PF10	F10	DOIA	Ctrl – X
PF11	F11	JOBCTRL	Ctrl – Z
PF12	F12		

Using e789 on DEC VT100 Keyboards

The following chart shows the default IBM AIX 3270 Host Connection Program/6000 key functions on the DEC VT100 terminal keyboard. See e789 Special Keys on page 4–3 for a description of the functions.

The DEC VT100 must be configured with the keypad in application mode for the **e789** process to function properly. This can be accomplished by invoking the following command on the AIX command line of the VT100 terminal:

```
echo "\033\075"
```

e789 Key Functions on DEC VT100 Keyboards			Part 1 of 2
Function	**Press**	**Function**	**Press**
TAB	TAB	PF13	N/A
BTAB	N/A	PF14	N/A
ENTER	Keypad ENTER	PF15	N/A
SYSREQ	Keypad 3	PF16	N/A
CLEAR	Keypad Period (.)	PF17	N/A
BACKSPACE	BACKSPACE	PF18	N/A
CHGDIS	N/A	PF19	N/A
INSERT	Keypad 0	PF20	N/A
DELETE	DELETE	PF21	N/A
HOME	N/A	PF22	N/A
PA1	Keypad 1	PF23	N/A
PA2	Keypad 2	PF24	N/A
PA3	N/A	PRINT	N/A
SPACEOF	N/A	SAVES	N/A
DOWN	Down Arrow	REPLS	N/A
DDOWN	N/A	BTABW	Ctrl – A
RIGHT	Right Arrow	EOF	Ctrl – B
RRIGHT	N/A	SHELL	Ctrl – C
LEFT	Left Arrow	QUIT	Ctrl – D
LLEFT	N/A	ERASE	Ctrl – E

e789 Key Functions on DEC VT100 Keyboards			Part 2 of 2
Function	**Press**	**Function**	**Press**
UP	Up Arrow	FTABW	Ctrl – F
UUP	N/A	ENDF	Ctrl – G
NEWLINE	RETURN	BACKF	Ctrl – K
PF1	PF1	LENDF	Ctrl – L
PF2	PF2	NEWLINE	Ctrl – M
PF3	PF3	FMARK	Ctrl – N
PF4	PF4	NEXT	Ctrl – O
PF5	Keypad 7	POREST	Ctrl – P
PF6	Keypad 8	RESET	Ctrl – R
PF7	Keypad 9	REFRESH	Ctrl – T
PF8	Keypad Minus (–)	DUP	Ctrl – U
PF9	Keypad 4	DELWORD	Ctrl – W
PF10	Keypad 5	DOIA	Ctrl – X
PF11	Keypad 6	JOBCTRL	Ctrl – Z
PF12	Keypad Comma (,)		

Using e789 on DEC VT220 Keyboards

The following chart shows the default IBM AIX 3270 Host Connection Program/6000 key functions on the DEC VT220 terminal keyboard. See e789 Special Keys on page 4–3 for a description of the functions.

The DEC VT220 must be configured as follows, for the **e789** process to function properly:

- VT220, 7–bit mode
- Keypad in application mode.

e789 Key Functions on DEC VT220 Keyboards			Part 1 of 2
Function	**Press**	**Function**	**Press**
TAB	Tab	PF13	F13
BTAB	N/A	PF14	F14
ENTER	Keypad 0	PF15	Help
SYSREQ	Keypad 1	PF16	Do
CLEAR	Keypad Period (.)	PF17	F17
BACKSPACE	Backspace	PF18	F18
CHGDIS	N/A	PF19	F19
INSERT	Find	PF20	F20
DELETE	Select	PF21	Keypad 6
HOME	Insert Here	PF22	Keypad 7
PA1	Remove	PF23	Keypad 8
PA2	Next Screen	PF24	Keypad 9

e789 Key Functions on DEC VT220 Keyboards			Part 2 of 2
Function	**Press**	**Function**	**Press**
PA3	N/A	PRINT	Keypad Minus (–)
SPACEOF	Prev Screen	SAVES	Keypad 4
DOWN	Down Arrow	REPLS	Keypad Comma (,)
DDOWN	N/A	BTABW	Ctrl – A
RIGHT	Right Arrow	EOF	Ctrl – B
RRIGHT	N/A	SHELL	Ctrl – C
LEFT	Left Arrow	QUIT	Ctrl – D
LLEFT	N/A	ERASE	Ctrl – E
UP	Up Arrow	FTABW	Ctrl – F
UUP	N/A	ENDF	Ctrl – G
NEWLINE	Return	BACKF	Ctrl – K
PF1	PF1	LENDF	Ctrl – L
PF2	PF2	NEWLINE	Ctrl – M
PF3	PF3	FMARK	Ctrl – N
PF4	PF4	NEXT	Ctrl – O
PF5	Keypad 5	POREST	Ctrl – P
PF6	F6	RESET	Ctrl – R
PF7	F7	REFRESH	Ctrl – T
PF8	F8	DUP	Ctrl – U
PF9	F9	DELWORD	Ctrl – W
PF10	F10	DOIA	Ctrl – X
PF11	F11	JOBCTRL	Ctrl – Z
PF12	F12		

Using e789 on WYSE WY–50 Keyboards

The following chart shows the default IBM AIX 3270 Host Connection Program/6000 key functions for the WYSE WY–50 terminal keyboard. See e789 Special Keys on page 4–3 for a description of the functions.

The WYSE WY–50 must be configured as follows, for the **e789** process to function properly:

- WY–50 Compatible Mode

e789 Key Functions on WYSE WY–50 Keyboards			Part 1 of 2
Function	**Press**	**Function**	**Press**
TAB	Tab	PF13	F13
BTAB	Shift – Tab	PF14	F14
ENTER	Send	PF15	F15
SYSREQ	CLR Scrn	PF16	F16

e789 Key Functions on WYSE WY–50 Keyboards			Part 2 of 2
Function	**Press**	**Function**	**Press**
CLEAR	CLR Line	PF17	N/A
BACKSPACE	Del	PF18	N/A
CHGDIS	N/A	PF19	N/A
INSERT	INS Char	PF20	N/A
DELETE	DEL Char	PF21	N/A
HOME	Home	PF22	N/A
PA1	PAGE Next	PF23	N/A
PA2	Prev PAGE	PF24	N/A
PA3	INS Line	PRINT	Print
SPACEOF	DEL Line	SAVES	Ins
DOWN	Down Arrow	REPLS	Repl
DDOWN	N/A	BTABW	Ctrl – A
RIGHT	Right Arrow	EOF	Ctrl – B
RRIGHT	N/A	SHELL	Ctrl – C
LEFT	Left Arrow	QUIT	Ctrl – D
LLEFT	N/A	ERASE	Ctrl – E
UP	Up Arrow	FTABW	Ctrl – F
UUP	N/A	ENDF	Ctrl – G
NEWLINE	Return	BACKF	Ctrl – K
PF1	F1	LENDF	Ctrl – L
PF2	F2	NEWLINE	Ctrl – M
PF3	F3	FMARK	Ctrl – N
PF4	F4	NEXT	Ctrl – O
PF5	F5	POREST	Ctrl – P
PF6	F6	RESET	Ctrl – R
PF7	F7	REFRESH	Ctrl – T
PF8	F8	DUP	Ctrl – U
PF9	F9	DELWORD	Ctrl – W
PF10	F10	DOIA	Ctrl – X
PF11	F11	JOBCTRL	Ctrl – Z
PF12	F12		

Understanding the HCON Operator Information Area (OIA)

The last line on the display screen while using the IBM AIX 3270 Host Connection Program/6000 is the Operator Information Area (OIA). It contains the following information:

Column	Description
1 to 3	Readiness and System Connection Information
9 to 17	Input Inhibited Field Information Input Inhibited Field Error Codes
21 to 27	Reminders – Comm. Check only. 603 Fatal Comm Check (Communication Line Problems)
38	Insert Mode Field Information
51 to 56	Internal Error Indicator Field Information
58	Session Name Field Information
59	Colon Separator
60 to 79	Session Use Field Information.

Readiness and System Connection Information

The readiness and system connection field occupies columns 1 to 3 of the Operator Information Area (OIA). The following table contains a list of the symbols, the column in which they start, and the function of the symbols used in this field:

OIA: Readiness and System Connection Fields		
Column	**Symbol**	**Description**
1	4	3274 Control Unit and session are ready.
1	S	3174 Control Unit and session are ready.
1	I	4361 Integrated Work Station Adapter and session are ready.
1	N	9370 Work Station Subsystem Controller and session are ready.
1	8	5088 Graphics Control Unit and session are ready.
2	A	The control unit is connected to the host under rule A.
3	"	The session is working with the host application.

Input Inhibited Field Information

The input inhibited field occupies columns 9 to 17 of the Operator Information Area (OIA). The following table contains a list of the symbols, the column in which they start, and the function of the symbols used in this field:

OIA: Input Inhibited Fields		
Column	**Symbol**	**Description**
9	X ( )	The host system needs time to perform a function.
9	X SYSTEM	The host system locked the keyboard.
9	X MACH*nnn*	Hardware problem. The machine check number will be in the ranges of 201–269, 290–299, and 601–699. (This is equivalent to the 3278/79 error indicators X back–slash box, and back–slash box.)
9	X – +z_*nnn*	Communication line problem. The communication number will be in the range of 501–599. (This is equivalent to the 3278/79 error indicator X broken communication symbol.)
9	X PROG*nnn*	A programming error has been detected in the data received from the host. The error number will be in the ranges of 401–499, and 701–799. (This is equivalent to the 3278/79 error indicator X PROG.)
9	X ?+	The last input is not accepted.
9	X –f	The requested function is not available.
9	X < 0 >	The cursor must be moved. It is in a protected field and an attempt has been made to enter, insert, erase, or delete a character.
9	X 0 >	An attempt has been made to enter more data into a field than can fit into it.
9	X –S	The symbol entered is not available.

Note: Selected Input Inhibited Field Error Codes are defined beginning on page 4–15.

Insert Mode Field Information

The mode field occupies column 38 of the Operator Information Area (OIA). The following table contains the symbol, the column in which it starts, and the function of the symbol:

OIA: Insert Mode Field		
Column	**Symbol**	**Description**
38	^	Insert mode is on.

Internal Error Indicator Field Information

The internal error indicator field occupies columns 51 to 56 of the Operator Information Area (OIA). This field notifies the operator that an error occurred during the operation of **e789**. The following table contains the symbol, the column in which it starts, and the function of the symbol:

OIA: Internal Error Indicator Field		
Column	**Symbol**	**Description**
51	ERR*nnn*	Indicates an internal emulator error. *nnn* is the error number. Record the error number and call your IBM representative.

Session Name Field Information

The session name field occupies column 58 of the Operator Information Area (OIA). The following table contains the symbol, the column in which it starts, and the function of the symbol:

OIA: Session Name Field		
Column	**Symbol**	**Description**
58	a through z	The session name which is specified in the session profile and is used to identify the session.

Session Use Field Information

The session use field occupies columns 60 to 79 of the Operator Information Area (OIA). The following table contains the symbol, the column in which it starts, and the function of the symbol:

OIA: Session Use Field		
Column	**Symbol**	**Description**
60	session use	A character string which is specified in the session profile and is used to identify the session.

Input Inhibited Field Error Codes

The following table lists possible Input Inhibited Field Error Codes, an explanation of the problem, and user responses:

OIA: Input Inhibited Field Error Codes		Part 1 of 3
Error Code	**Explanation**	**User Response**
670	Hardware timeout	Please record the information and call your IBM representative.
671	Device Initialization error	Please record the information and call your IBM representative.
674	Hardware error	Please record the information and call your IBM representative.
706	Command chaining error	Press the **RESET** key to clear the error. If the problem continues, record the information and contact the host system programmer.

OIA: Input Inhibited Field Error Codes		Part 2 of 3
Error Code	**Explanation**	**User Response**
706	Read Partition was received while the device was in retry state	Press the **RESET** key to clear the error. If the problem continues, record the information and contact the host system programmer.
750	Invalid 3270 command	Press the **RESET** key to clear the error. If the problem continues, record the information and contact the host system programmer.
752	Address out of range	Press the **RESET** key to clear the error. If the problem continues, record the information and contact the host system programmer.
755	Invalid character set	Press the **RESET** key to clear the error. If the problem continues, record the information and contact the host system programmer.
756	SF type invalid or unsupported	Press the **RESET** key to clear the error. If the problem continues, record the information and contact the host system programmer.
759	Error in length given for SF	Press the **RESET** key to clear the error. If the problem continues, record the information and contact the host system programmer.
761	Wrong pid value for this SF	Press the **RESET** key to clear the error. If the problem continues, record the information and contact the host system programmer.
769	HCON internal error	Press the **RESET** key to clear the error. If the problem continues, record the information and contact the host system programmer.
770	HCON internal error	Please record the information and call your IBM representative.
771	HCON internal error	Please record the information and call your IBM representative.
772	HCON internal error	Please record the information and call your IBM representative
773	HCON internal error	Please record the information and call your IBM representative.

OIA: Input Inhibited Field Error Codes		Part 3 of 3
Error Code	**Explanation**	**User Response**
774	HCON internal error	Please record the information and call your IBM representative.
775	HCON internal error	Please record the information and call your IBM representative.
776	HCON internal error	Please record the information and call your IBM representative.
777	Device address already enabled	Please record the information and call your IBM representative.
778	Device address not enabled	Please record the information and call your IBM representative.
779	HCON internal error	Please record the information and call your IBM representative.
780	HCON internal error	Please record the information and call your IBM representative.
781	HCON internal error	Please record the information and call your IBM representative.
782	HCON internal error	Please record the information and call your IBM representative.
783	Selective reset issued	The host has issued a selective reset. Consult your local host administrator.
784	Power on reset in progress	Press the **RESET** key to clear the error. Continue with the host session.
785	Command reject generated by 5088	Press the **RESET** key to clear the error. Continue with the host session.
786	5088 detected operation check	Press the **RESET** key to clear the error. Continue with the host session.
787	The Adapter has shut down	Please record the information and call your IBM representative.
799	HCON internal error	Please record the information and call your IBM representative

Other Possible Errors

The preceding list of error codes represents the events most likely to occur. It is not a complete list of all possible 3270 errors. The complete list is in *IBM Control Unit Reference Summary* documentation.

Related Information

The **e789** command, **e789cdef** command, **e789cln** command, **e789kdef** command, **fxfer** command, **mkhcons** command.

HCON Introduction for System Management

The AIX 3270 Host Connection Program/6000 (HCON) is a software package that allows a RISC System/6000 to communicate with one or more IBM System/370 computer systems. Conceptual information about managing HCON is divided into four groups of information:

- Read the HCON Overview for System Management on page 4–19 to find out about the parts of HCON and how they work on your system.
- Read the HCON Overview of System Management Tasks on page 4–33 to find out about the tasks that need to be completed so that HCON can be used on your system.
- Read the HCON Host Configuration Overview on page 4–42 to find out how to set up a System/370 host so that it can be used with HCON.
- Read the List of HCON System Management Information on page 4–50 for a list of conceptual, procedural, and reference information about managing HCON.

For information about running HCON, read the HCON Overview on page 4–1. For information about the HCON Applications Programming Interface (API), the HCON File Transfer Programming Interface, and explicit and implicit logon procedures, read the HCON Overview for Programming in *Communications Programming Concepts*.

HCON Overview for System Management

The AIX 3270 Host Connection Program/6000 (HCON) is a software package that allows a RISC System/6000 to communicate with one or more IBM System/370 computer systems. To communicate with a System/370 host, HCON establishes a *session*, which is a period of activity with the host system.

HCON can be used to establish multiple sessions with IBM System/370 computers. Each session emulates either a 3278/79 display or a 3286/87 printer. A session emulating a display is known as a *display session*, while a session emulating a printer is known as a *printer session*. HCON provides file transfer capabilities within display sessions. It also includes an Application Programming Interface (API), which can be used to write programs that communicate with a System/370 host.

Each user of HCON can have up to 26 sessions, allowing one or more simultaneous invocations of HCON to communicate with one or more hosts using different session characteristics and communication protocols. The parameters defining the characteristics of each session are established by a *session profile*.

HCON uses the 3270 Connection Adapter in Distributed Function Terminal (DFT) mode or the System/370 Host Interface Adapter (HIA) to communicate with a System/370. If your RISC System/6000 has both types of devices installed, users can use either device for different HCON sessions. Sessions using the 3270 Connection Adapter are known as *non-SNA DFT sessions*. Sessions using the HIA are known as *HIA sessions*.

The 3270 Connection Adapter can be used to emulate a display by establishing a *non-SNA DFT display session*, or a printer by establishing a *non-SNA DFT printer session*. The HIA adapter can be used to emulate a display by establishing an *HIA display session*.

Your system can have any combination of one to four 3270 Connection Adapters or one HIA. Each 3270 Connection Adapter can support up to five sessions while the HIA can support up to 16 sessions. Thus the maximum number of simultaneous sessions which can run on your system is 36 (20 non-SNA DFT sessions and 16 HIA sessions), if you have all of the necessary hardware installed and have host connections configured and available.

For further information about the parts of HCON and how they work on the RISC System/6000, read Understanding HCON Users on page 4–20, Understanding HCON Sessions on page 4–20, Understanding HCON Processes on page 4–29, and Understanding the HCON File Transfer Process on page 4–31.

For information about the tasks that need to be completed so that HCON can be used on your system, read the HCON Overview of System Management Tasks on page 4–33.

For a list of information in the database that is especially useful when managing HCON, read the List of HCON System Management Information on page 4–50.

Related Information

HCON Host Configuration Overview on page 4–42 describes the configuration requirements for a System/370 host that is to be used with HCON.

HCON Overview for Programming in *Communications Programming Concepts* discusses programming with the HCON API and File Transfer Programming Interface.

Understanding HCON Users

Before you can use the AIX 3270 Host Connection Program/6000 (HCON), you must be registered to HCON as an HCON user. An HCON user can define HCON session profiles and then use HCON to:

- Establish display or printer sessions with a System/370 computer
- Use the HCON file transfer facility (the **fxfer** command) to transfer files
- Run programs written with the HCON API.

To define a new user to HCON, you must have root user authority. A root user can define an HCON user with the **smit hcon** command or with the **mkhconu** command. To add or remove an HCON user, specify the login name of the user to be added or removed. When you add a user, the name you specify must be a valid AIX login name. When you remove a user, the name you specify must be that of a registered HCON user.

HCON uses an Object Data Manager (ODM) database to store information about HCON users. When an HCON user is registered, an entry for that user is created in the ODM database, using the HCON system defaults for session information. Thereafter, each time the user creates a new session, HCON changes the user's defaults in the ODM database to match the values set for the last session the user created.

For further information about creating HCON users, read Configuring HCON on page 4–35.

For further information about sessions, session profiles, and file transfers, read Understanding HCON Sessions on page 4–20 and Understanding the HCON File Transfer Process on page 4–31.

Related Information

Use the **smit hcon** command to configure HCON with SMIT.

The root user can define an HCON user with SMIT or the **mkhconu** command, view a list of HCON users with SMIT or the **lshconu** command, and remove HCON users with SMIT or the **rmhconu** command.

The **e789** command, **e789cln** command, **e789cdef** command, **e789kdef** command, and **fxfer** command can be issued by HCON users.

Object Data Manager (ODM) Overview in *General Programming Concepts* discusses how data is stored in the ODM, which is used to store HCON user information.

Understanding HCON Sessions

An HCON session is a period of activity with an IBM System/370 host computer. The HCON **e789** command, **fxfer** command, and Application Programming Interface (API) provide the means to establish or connect to HCON sessions.

Depending on the hardware devices installed on the system, up to 36 sessions of HCON can be running simultaneously (but with no more than 26 sessions per HCON user).

Each session has a session profile. Before invoking the HCON emulator or file transfer, or programs written with the API, each user must create at least one session profile. Then, when invoking one of the functions of HCON, the user specifies which session profile to use.

Display Sessions

Display sessions allow users to emulate a 3278/79 display terminal connected to an IBM System/370 host computer. At the beginning of the session, it is as if the terminal had just been turned on. The user can log on to the host, run programs on the host, and transfer files between the RISC System/6000 and the host.

HCON supports two types of display sessions, non-SNA DFT sessions and HIA sessions. These sessions differ primarily in the type of device used to connect to the System/370 host computer. Non-SNA DFT display sessions use the 3270 Connection Adapter in Distributed Function Terminal (DFT) mode. HIA display sessions use the System/370 Host Interface Adapter (HIA).

Printer Sessions

An HCON printer session allows a user to emulate a 3286/87 printer. With a printer session, an HCON user can print files from the host computer on a printer attached to the RISC System/6000, or store these files in printer form in a file on the RISC System/6000.

HCON supports one type of printer session, the non-SNA DFT printer session. This session uses the 3270 Connection Adapter in Distributed Function Terminal (DFT) mode to connect to the host computer.

For further information about session profiles and their characteristics, read Understanding HCON Session Profiles on page 4–21.

For information about steps you can take if a session is interrupted, read Working with Interrupted HCON Sessions on page 4–40. For information about file transfers, read Understanding the HCON File Transfer Process on page 4–31.

Related Information

The **e789** command, **fxfer** command, and HCON API use HCON sessions and session profiles. Understanding the AIX Interface for HCON in *Communications Programming Concepts* discusses how the HCON API uses sessions.

Understanding HCON Session Profiles

An HCON session profile establishes certain defaults that affect the way HCON runs. Each HCON user can have up to 26 session profiles, each designated by a lowercase letter of the alphabet.

Session profiles are stored using the Object Data Manager (ODM). The ODM stores data in *object classes*. Associated with each HCON user are two object classes. The object classes consist of four files in the user's **$HOME** directory. These files are created when a user is registered as an HCON user. Session profiles for the user are stored in the **$HOME/usrprofs** and **$HOME/usrprofs.vc** object class files. The defaults for that user's session profiles are stored in the **$HOME/usrdflts** and **$HOME/usrdflts.vc** object class files. Global defaults for HCON sessions are stored in object class files in the **/usr/lib/hcon** directory, along with a list of registered HCON users.

When a session profile is created, the data specified for the profile is stored in the **usrprofs** object class. Values for options that were not specified are taken from the **usrdflts** object class. Then, the new session profile is used to update the **usrdflts** object class. Thus the values specified for the new profile become the user's defaults for future profiles of that type.

Session profiles can be created and modified with the **smit hcon** command or with the **mkhcons** and **chhcons** commands. To list existing session profiles from the command line, use the **lshconp** command. To list the parameters of an existing session profile from the

command line, use the **lshcons** command. To remove a session profile from the command line, use the **rmhcons** command. (You can also use the **smit hcon** command to perform these functions.) To find out the session type of an existing profile, use the **clhcons** command.

A session profile can define either a printer session or a display session. Each profile contains a number of characteristics, depending on the type of session the profile defines. Each characteristic, other than the session type, has an associated field in the SMIT session profile dialogs. In the **mkhcons** and **chhcons** commands, each profile characteristic is associated with a flag.

A SMIT dialog shows only the characteristics appropriate to the type of session you are creating. If you use the **mkhcons** or **chhcons** command, however, specifying a printer session characteristic when defining a display session generates an error message. Likewise, specifying a display session characteristic when defining a printer session generates an error message.

Following are descriptions of the characteristics common to all session profiles, characteristics for display session profiles, and characteristics for printer session profiles.

Characteristics Common to All Session Profiles

Session profiles for both display and printer sessions define the following:

- Name of the user to whom the session belongs
- Session type
- Session name and session use
- Type of communication device to be used
- Files to use for color and keyboard definitions
- National language table to use for ASCII–EBCDIC translation.

HCON User Name

The HCON user name specifies the AIX login name of the owner of the session to be shown, created, modified, or removed. The login name specified must be that of a registered HCON user. In SMIT, the default is shown in the dialog field. With the session profile commands such as **mkhcons** and **lshcons**, the user name defaults to the current login name.

Session Type

The session type specifies whether the profile is for a non-SNA DFT printer session, a non-SNA DFT display session, or an HIA display session. Once the session profile has been created, the session type cannot be changed.

When you create a profile using SMIT, you can choose from a list of available session types. The list differs depending on how you entered the HCON session configuration dialog. If you entered it from an adapter configuration menu, only the types of sessions for that adapter are shown. If you entered the dialog from the HCON configuration menu, all session types are shown.

When using the **mkhcons** command to create a session profile, specify the session type with the **–t** flag. To find out the type of an existing session profile from the command line, use the **clhcons** command or the **lshcons** command.

Session Name

Each HCON session has a unique name, which is a single character in the range of a through z. Capital letters are interpreted as lowercase letters. When you define a session, the name you specify must not be the name of a previously defined session profile for the user.

The session name must be specified. Once the session has been defined, the session *name* cannot be changed. However, sessions that already exist can be reconfigured.

In SMIT, the session name for a new session defaults to the first available letter in the range of a through z. There is no default session name for other options, such as reconfigure or remove. In addition, there is no default session name for profiles created without SMIT.

The **mkhcons** command does not provide a default for the session name. When defining a session with the **mkhcons** command at the command line, you must specify a session name with the **–n** flag. Use the **lshconp** command to find out which session names have already been used.

Session Use

The session use, which is optional, is a string of up to 20 ASCII characters. It is intended to help identify the session profile. The string you enter for the session use is shown in the Operator Information Area (OIA) when a session is running, and in lists of sessions produced using either SMIT or the **lshconp** command. The default for the session use is blank.

Communication Device

This characteristic specifies the communication device name, which can be up to 10 characters long. For non-SNA DFT display and printer session profiles, the device name is the IBM 3270 Connection Adapter device name (**3270c***X*, where *X* can be one of the numerals 0 through 7). For HIA display session profiles, the device name is **hia0**.

The communication device is defined either automatically during the first IPL after the device has been plugged into the system or manually using the **smit devices** command. A communication device does not have to be defined to be specified in a session profile. However, it must be defined before the HCON session can be run using the device name specified.

The communication device may not be omitted. Its default is **3270c0** for non-SNA DFT sessions or **hia0** for HIA sessions.

Country

The country indicates the code page to be used for ASCII–EBCDIC translation. The default value is determined by the value of the **LANG** environment variable at the time the user is registered as an HCON user. If the **LANG** variable is not set or has an invalid value, United States is used as the default.

The following values are valid for the country:

Code Pages for ASCII–EBCDIC Translation		
Country	**Value of the LANG variable**	**Command line abbreviation**
United States	En_US	USA
German/Austria	Gr_GR	GER
Belgium	Du_BE, Fr_BE, Gr_BE	BEL
Denmark	Da_DK	DEN
Norway	No_NO	NOR
Finland	Fi_SU	FIN
Swiss French	Fr_SW	SWF
Swiss German	Gr_SW	SWG
Sweden	Sv_SV	SWE
Italy	It_IT	ITA
Japan (Latin)	Jp_JP	JAP
Portugal	Po_PO	POR
Spain / Latin America	Sp_SP	SPA
United Kingdom	En_UK	UK
France	Fr_FR	FRA
Canada	Fr_CF	CAN

Keyboard Table

The keyboard table, which can be up to 1024 characters long, specifies the file containing the keyboard definition table to be used for the session. The specified table tells HCON where emulator keys are mapped on the keyboard. For example, it tells HCON which key on the keyboard should act like the Clear key found on a 3278/79 display. The table is built using the keyboard definition command, **e789kdef**. The default table is the **/usr/lib/hcon/e789_ktbl** file, which is installed with HCON.

Color Table

The color table, which can be up to 1024 characters long, specifies the file containing the color definition table to be used for the session. The specified table tells the emulator the colors and attributes to be used for each 3270 field and character attribute. The table is built using the color definition command, **e789cdef**. The default table is the **/usr/lib/hcon/e789_ctbl** file, which is installed with HCON.

Characteristics for Display Session Profiles

For a display session, the session profile also defines:

- Type of operating system on the host computer
- Type of 3270 model to emulate
- Login ID on the System/370 host
- Names of files in which the emulator saves the host display when the SAVES and REPLS functions are used
- Name of the file transfer program on the System/370 host
- Default direction for file transfers (upload to the System/370 or download from the System/370)
- Length of time the file transfer program waits for more transfer requests before logging off from the host or disconnecting from the emulator session
- Length of time the file transfer program attempts to recover a failed file transfer
- AUTOLOG node ID, AUTOLOG trace, and AUTOLOG timeout for implicit logons
- Communication buffer size for file transfer and API.

Host Type

The host type characteristic specifies the type of operating system used by the System/370 host computer. It is used by the file transfer program and the API. The following values are valid:

TSO System/370 is running an MVS/TSO operating system (default).

CMS System/370 is running a VM/CMS operating system.

3270 Model (Rows x Columns)

This characteristic specifies the 3270 model to be emulated. The number of rows and columns is also displayed. Following are the valid values:

2 3278/79 model 2: screen size is 24 rows by 80 columns (default)

3 3278/79 model 3: screen size is 32 rows by 80 columns

4 3278/79 model 4: screen size is 43 rows by 80 columns

5 3278/79 model 5: screen size is 27 rows by 132 columns.

Note that the actual emulator screen contains one more row than indicated in the preceding list. The extra row contains the Operator Information Area (OIA).

Except on ASCII terminals, the emulator will not bring up a session unless the entire emulator screen fits on the physical display (or window). Only model 2 sessions will run on ASCII terminals. In such sessions, either row 24 or the OIA is displayed on the bottom row of the display, and the operator can switch back and forth between them.

If the display (or window) is larger than that of the specified model, the OIA is displayed on the last row, and the emulator screen is displayed in the upper left-hand corner of the physical screen (or window).

File Used by SAVES Key

This characteristic defines the save file name, which can be up to 1024 characters long. Each time the SAVES key is pressed, the current screen is appended to the file identified by

the save file name. The actual file name is the specified name concatenated with a period and the session name. For example, if `/u/rps/save` is specified as the file name and the name of the session is `a`, screen images are appended to the `/u/rps/save.a` file. The default name of the save file is the **$HOME/e789_save** file.

File Used by REPLS Key

This characteristic defines the replace file name, which can be up to 1024 characters long. Each time the REPLS key is pressed, the current screen is used to replace the contents of the file identified by the replace file name. The image replaces any data in the file, including previously saved screen images. The actual file name is the specified name concatenated with a period and the session name. For example, if `/u/rps/replace` is specified as the file name and the name of the session is `a`, the screen image is placed in the `/u/rps/replace.a` file. The default name of the replace file is the **$HOME/e789_repl** file.

Host Login ID

The host login ID specifies a System/370 login name of up to eight characters to be used by the file transfer program and API for implicit logons using the AUTOLOG or LAF facility. The host login ID may be left blank, which is the default.

Host File Transfer Program

The host file transfer program can be up to 8 characters long and specifies the System/370 file transfer program name. The default depends on the country specified by the value of the **LANG** environment variable at the time the user is registered as an HCON user. If the **LANG** variable is not set or has an invalid variable, the **IND$FILE** program is used as the default.

Following are the default values for the host file transfer program for each supported value of the **LANG** variable:

Host File Transfer Programs for Specific Countries		
Country	**Value of the LANG variable**	**Default host file transfer program**
United States	En_US	IND$FILE
German/Austria	Gr_GR	IND$FILE
Belgium	Du_BE, Fr_BE, Gr_BE	IND$FILE
Denmark	Da_DK	INDÅFILE
Norway	No_NO	INDÅFILE
Finland	Fi_SU	INDÅFILE
Swiss French	Fr_SW	IND$FILE
Swiss German	Gr_SW	IND$FILE
Sweden	Sv_SV	INDÅFILE
Italy	It_IT	IND$FILE
Japan (Latin)	Jp_JP	IND¥FILE
Portugal	Po_PO	IND$FILE
Spain / Latin America	Sp_SP	IND$FILE
United Kingdom	En_UK	IND£FILE
France	Fr_FR	IND$FILE
Canada	Fr_CF	IND$FILE

File Transfer Direction

This characteristic specifies the default file transfer direction. Its value should indicate the direction most often used during file transfer. The default value is down.

File Transfer Wait Period

The file transfer wait period specifies the amount of time, in minutes, that the file transfer **dfxfer** process waits for more file transfer requests before logging off from the host computer or disconnecting from the emulator session. If the **dfxfer** process is running in implicit mode, the associated session also remains logged on to the host during this period. If the wait period ends and no requests have been received, the **dfxfer** process disconnects from the session if running explicitly, or logs off the host and terminates the session if running implicitly.

The wait period must be a number of minutes up to 999. A value of 0 (zero) is the default and indicates that the **dfxfer** process should exit immediately after completing the last file transfer.

A value of 999 indicates that the **dfxfer** process should never terminate. For implicit file transfers, a value of 999 causes the associated session to remain logged on to the host indefinitely, or until it is terminated.

File Transfer Recovery Time

The file transfer recovery time specifies the length of time, in minutes, that the file transfer **dfxfer** process should attempt recovery during file transfer. The recovery time must be a number of minutes up to 999. The recovery time is only in effect when the file transfer process has performed an implicit logon to the host. A value of 0 (zero) is the default and indicates that no recovery should be performed.

AUTOLOG Node ID

The AUTOLOG node ID, which can be up to 18 characters, specifies the AUTOLOG profile to be used by the AUTOLOG facility. An AUTOLOG profile must be created before an implicit logon using AUTOLOG is attempted by file transfer or an API program. The AUTOLOG node ID may be left blank, which is the default. Leaving it blank when there is a Host Login ID specified indicates that LAF is the preferred implicit logon procedure.

AUTOLOG Trace

The AUTOLOG trace specifies whether a trace should be sent to standard output during an implicit host logon using AUTOLOG. The trace contains a listing of the AUTOLOG procedure followed by the status of each step executed during the logon process. If you choose `yes`, a trace is issued during implicit logons performed with AUTOLOG. The default value is `no`.

AUTOLOG Timeout

The AUTOLOG timeout, which can be up to 99 seconds, specifies the maximum amount of time, in seconds, that the AUTOLOG facility waits for a specified prompt to be sent from the host system.

If the specified prompt is received before the timeout period is complete, the AUTOLOG facility immediately sends the specified input to the host and proceeds to the first of the next events. If the timeout period is completed without receiving the prompt, the AUTOLOG facility proceeds to the next unsuccessful event number. A value of 0 (zero), which is the default, indicates that the next step should be performed immediately.

Maximum I/O Buffer Size (Bytes)

The I/O buffer size specifies the maximum size, in bytes, of the session's communication buffer used during file transfer and API/API or API/API_T operations.

If the number specified for the I/O buffer size is larger than the buffer size specified during the configuration of the communication device, the specified maximum size is ignored and the communication buffer size is used instead.

The maximum value that can be given for the I/O buffer size is 32767 bytes. The default for the maximum I/O buffer size is 3456 bytes, which is the largest value that can be used when running a file transfer, API/API, or API/API_T operation through the **pvm** or **passthru** facility on the host system.

Warning: If the default of 3456 bytes is not used while the operation is going through the **passthru** or **pvm** facility, unknown results can occur, especially concerning PROG checks on the emulator.

Characteristics for Printer Session Profiles

For a printer session, the session profile defines:

- Local printer name
- Name of the local file in which to store printer output
- Controller port address.

Printer Local Name

The three-character printer local name specifies the name of the local printer used by the emulator to service print requests from the host. The printer need not have been previously set up before being specified in a session profile. However, it must be set up before the HCON emulator is invoked with the session you are defining. The printer local name defaults to `lp0`.

The local printer name can be left blank. Either a local printer name or a file to store printer output must be specified, but not both.

File To Store Printer Output

The file to store printer output, which can be up to 1024 characters long, specifies the name of the file in which to store printer output instead of spooling it to a physical printer.

The printer output file can be left blank, which is the default. Either a printer output file or a printer local name must be specified, but not both.

Controller Port Address

The two-character controller port address specifies the logical port number where the coaxial cable from the 3270 Connection Adapter is attached to the 3174 or 3274 Control Unit, 4361 Work Station Adapter, or 9370 Work Station Subsystem Controller.

The controller port address entry is the last five low-order bits of the network or terminal address used by the host. The bits are represented by a two-character hexadecimal number in the range of 00 to 1F. The 3270 Connection Adapter printer address must also be configured with the same hexadecimal number. This can be done using the **smit devices** command. The controller port address may not be left blank; the default value is 00.

For further information about setting up session profiles, read Configuring HCON on page 4–35.

Related Information

To work with session profiles using SMIT, run the **smit hcon** command.

To set up a session profile for HCON, use SMIT or the **mkhcons** command. To modify an existing profile, use SMIT or the **chhcons** command.

To list existing session profiles, use SMIT or the **lshconp** command. To list the parameters of an existing session profile, use SMIT or the **lshcons** command. To find out the session type of an existing profile, use the **clhcons** command.

To remove a session profile, use SMIT or the **rmhcons** command.

Understanding HCON Explicit and Implicit Logon in *Communications Programming Concepts* discusses AUTOLOG and LAF.

Understanding the HCON API in *Communications Programming Concepts* discusses how the API uses sessions and session profiles.

Object Data Manager (ODM) Overview in *General Programming Concepts* discusses the ODM database, which is used to store HCON user and session profile information.

Understanding HCON Processes

The HCON emulator consists of a parent process (the **e789** process) and child processes (the **e789x** processes). Each time the **e789** command is invoked, an **e789** process is started. The **e789** process then starts an instance of the **e789x** process for each session.

For example, the command `e789 bdh` starts one instance of the **e789** process and three instances of the **e789x** process, one each for sessions `b`, `d`, and `h`.

Although each instance of the **e789** process manages the **e789x** processes associated with it, each invocation of the **e789** process runs independently of the other instances of the **e789** process which may have been invoked on the same system.

The **hcondmn** subsystem provides a systemwide focal point to coordinate the various HCON sessions and negotiate conflicts. The subsystem assigns a unique identifier to each session, providing the **e789**, **e789cln**, **fxfer**, and HCON API programs with a label identifying that session on the RISC System/6000.

In addition, the HCON file transfer starts a separate child process. This process, the **dfxfer** process, is invoked whenever a file transfer is begun by either the **fxfer** command, the **fxfer** function, or the **g32_fxfer** function.

The hcondmn Subsystem

The **hcondmn** subsystem is a process that is invoked at IPL time on any system where the HCON program has been installed. The HCON install process places instructions in the **inittabs** file so that the System Resource Controller (SRC) starts the **hcondmn** subsystem each time the system is turned on. The subsystem can also be started using the **sthcondmn** command.

The subsystem remains in a wait state until requests from HCON internal components arrive. Any process that attempts to establish an HCON session communicates with the **hcondmn** subsystem to register the session. The **hcondmn** subsystem:

- Resolves session name conflicts at session startup.
- Maintains and manages internal software structures such as queues, shared segments, and logical terminal names for each HCON session on the system.
- Makes interfaces to the sessions available on request of HCON applications so that interaction with the session or sessions is possible.
- Removes the resources associated with each session when that session is terminated.

The **hcondmn** subsystem can be stopped and started using commands. To stop the subsystem, use the **stopsrc** command, as follows:

```
stopsrc -s hcon
```

Warning: The **hcondmn** subsystem should normally be stopped only under directions from IBM Customer Support or when installing or updating HCON software. Be sure all HCON activity has ceased before stopping the **hcondmn** subsystem.

To start the subsystem, use the **sthcondmn** command, as follows:

```
/usr/lib/hcon/sthcondmn
```

The e789 Process

One instance of the HCON **e789** process is invoked each time the emulator is started using the **e789** command.

The **e789** process contacts the **hcondmn** subsystem to establish the requested session or sessions. The **hcondmn** subsystem sets up and allocates the appropriate storage space for each session and passes a return code to the **e789** process. Then, the **e789** process creates an **e789x** child process for each HCON session.

The e789x Process

One instance of the **e789x** child process is started for each HCON session on the system. The **e789x** process can be started by the **e789** process, by an HCON file transfer executed with an implicit logon, or by an HCON API program executed with implicit logon. The **e789x** process attaches to the storage space that has been set up for that session by the **hcondmn** subsystem.

For further information about the **dfxfer** process, read Understanding the HCON File Transfer Process on page 4–31.

For information about sessions, read Understanding HCON Sessions on page 4–20.

Related Information

Processes Overview in *General Concepts and Procedures* discusses AIX processes.

The following commands invoke and stop HCON processes: the **fxfer** command, **e789** command, and **e789cln** command.

The **sthcondmn** command is used to start the **hcondmn** subsystem. The **stopsrc** command stops the subsystem.

Understanding the HCON File Transfer Process

The HCON file transfer program allows you to transfer files between a RISC System/6000 and a System/370 host. The host operating system may be either VM/CMS or MVS/TSO, with the corresponding version of the IBM 3270 File Transfer Program (**IND$FILE**) installed. The file may be transferred from a RISC System/6000 to a host (uploading) or from the host to the RISC System/6000 (downloading). The HCON file transfer supports transfer of either text or binary data.

The file transfer program consists of two modules, the **fxfer** process and the **dfxfer** process. The **fxfer** process can be started from the command line by the **fxfer** command. It can also be started by the **fxfer** function in a program using the HCON file transfer programming interface or by the **g32_fxfer** function in a program using the HCON API. The **fxfer** process interprets the file transfer requests, forms file transfer request queues, and sends the requests to the appropriate **dfxfer** process. The status of the file transfer operation can be reported to a terminal or a file (if invoked with the **fxfer** command), or to a program (if invoked by a program).

The **dfxfer** process transfers the files. Each **dfxfer** process is associated with one display session of HCON. Multiple file transfer requests using the same session are queued to the same **dfxfer** process. The **dfxfer** process can remain logged on to the host indefinitely, based on the file transfer wait period specified in the session profile.

File transfer may be invoked either synchronously or asynchronously. If the file transfer is asynchronous, the operator can start additional file transfers without waiting for the first one to complete. Each request is placed in the queue used by the **dfxfer** process that is associated with the appropriate display session. Up to 580 file transfers can be queued at any one time.

File transfers can use either implicit or explicit logons. When using an explicit logon, the user must establish a session with the host and log on before starting the file transfer.

Using implicit logon, the **fxfer** process checks for the appropriate session. If the session exists, the **fxfer** process connects to that session and starts the **dfxfer** process, which does the file transfer. If the session does not already exist on the RISC System/6000, the **fxfer**

process starts the **dfxfer** process, which then starts the session by starting an **e789x** process, logs on to the host, transfers the file, logs off the host, and terminates the session.

Note: If the requested session exists, it could have been started explicitly with the **e789** command, or implicitly by a previous file transfer.

The file transfer can be interrupted by the operator before it is completed. A file transfer may also be interrupted by any unrecoverable communication error. When interrupted, the state of the transfer is saved. The transfer can then be restarted without loss of data. See Working with Interrupted HCON File Transfers on page 4–39.

When users frequently perform similar file transfers to the same host, a shell procedure can be developed to specify the parameters and minimize user interaction with the file transfer process. How to Create HCON File Transfer Shell Procedures on page 4–63 discusses such procedures.

For information about restarting file transfers, read Working with Interrupted File Transfers on page 4–39..

Understanding the HCON File Transfer Programming Interface in *Communications Programming Concepts* discusses developing file transfer programs using the **fxfer** function. Understanding the HCON API in *Communications Programming Concepts* discusses setting up an API program using the **g32_fxfer** function.

Related Information

Understanding HCON Explicit and Implicit Logon in *Communications Programming Concepts* discusses implicit logon procedures.

How to Create HCON File Transfer Shell Procedures on page 4–63.

The **fxfer** command starts file transfers.

HCON Overview of System Management Tasks

Before HCON can be used on your system, the adapters that you are using to communicate with the System/370 must be installed and configured, and a physical connection to the System/370 must be set up. Additionally, HCON and the HCON Message Catalog (MRI) must be installed, and HCON must be configured. Installing and Updating HCON on page 4–33 and Configuring HCON on page 4–35 discuss setting up HCON on the RISC System/6000.

Also, each IBM System/370 with which a session is to be established must be correctly configured. Additionally, if the HCON API is to be used, it must be installed on the appropriate System/370 computer systems and made accessible to each login ID that will be using it. The HCON Host Configuration Overview on page 4–42 discusses host configuration requirements.

Typically, after a session is established with a System/370 computer, the user must log in. This is known as an *explicit logon*. However, HCON users can use automatic logons, or *implicit logons*, in conjunction with HCON file transfer or HCON API applications. For implicit logons, an automatic logon procedure must be defined using either the AUTOLOG facility or the Logon Assist Feature (LAF). Understanding HCON Explicit and Implicit Logon in *Communications Programming Concepts* discusses using the AUTOLOG and LAF facilities to log in to a host system.

HCON can be customized, allowing users to change the way 3270 field and character attributes are displayed and the way emulator keys are mapped on the keyboard. These changes are made by creating customized color and keyboard definition files and then specifying these files in a session profile. Customizing HCON on page 4–37 discusses changing HCON to fit individual preferences.

It may be necessary to change HCON configuration from time to time. For example, it may be necessary to add or delete HCON users or change session profiles. Occasionally, if a session or file transfer is interrupted (for example, because the connection with the host is lost), the HCON user or the root user may need to clean up the session's system resources and processes or restart the file transfer. Maintaining HCON on page 4–39 discusses these tasks.

For a list of information in the database that is especially useful when managing HCON, read the List of Information about HCON System Management on page 4–50 .

Installing and Updating HCON

Installing HCON

Before HCON can be run on a RISC System/6000, the following must be installed:

- HCON Software
- HCON Message Catalog (MRI)
- One or more adapters used to connect to a System/370 host.

HCON Software

The root user can install the HCON system by using either the System Management Interface Tool (SMIT) or the **installp** command. When the HCON installation process is run, it:

- Backs up any files that will be overwritten by the new installation.
- Installs the new files.
- Adds HCON configuration menus to the SMIT menus and HCON databases using the Object Data Manager (ODM).
- Adds commands to the System Resource Controller (SRC) to start the **hcondmn** subsystem at system startup.

If the installation was unsuccessful for any reason, the procedure removes any new files that have been installed and replaces them with the old files. (This does not, however, guarantee that the previous version of HCON will work.)

HCON MRI

In addition to installing HCON itself, you must also install the HCON message catalog, also known as the Machine Readable Information (MRI). The MRI provides HCON messages in the appropriate national language for your system. It is possible to install HCON messages for more than one national language. When multiple MRIs are installed, the first MRI installed on the system becomes the primary default MRI for HCON. To use another MRI, the user must set the **LANG** environment variable to the appropriate national language. The HCON MRI is usually installed at the same time as the HCON software.

Adapters for HCON

Before HCON can be run on your system, you must install and configure either a 3270 Connection Adapter or a System/370 Host Interface Adapter (HIA). You can install the adapter either before or after you install the HCON software and MRI, but you cannot run HCON until at least one adapter is installed. The adapters are installed and configured the first time you IPL your system after attaching the adapter hardware. Use SMIT to modify the adapter configuration as needed.

Next Steps

To complete setting up HCON, you must configure HCON on the RISC System/6000 and the System/370 host.

Note: Before using HCON API programs to contact a System/370 host, the HCON Host API must be installed on the System/370 host. To install it, you must be able to establish an HCON session with that host and log in. Thus, you must configure HCON on both the RISC System/6000 and the host before installing the host API.

Installing HCON in a Codeserver Environment

The HCON software can be installed on a codeserver and run from a client of the server. In a codeserver environment, the adapter used to communicate with the System/370 host must be installed on the client. In addition, the System Resource Controller (SRC) and the ODM database must be installed on the client. Finally, HCON configuration must be done on the client, not on the codeserver.

Updating HCON

Occasionally the HCON software needs to be updated. Use SMIT or the **updatep** command to update the software. The update procedure goes through the same series of steps as the install procedure for HCON software.

For further information about installing and updating HCON, read the procedures How to Install HCON on page 4–53 and How to Set Up HCON in a Codeserver Environment on page 4–58.

Related Information

Understanding the Host Interface for HCON API in *Communications Programming Concepts* discusses installing the Host API.

How to Install the HCON MVS/TSO Host API, How to Install the HCON VM/CMS Host API in *Communications Programming Concepts.*

The **installp** command and **updatep** command can be used to install and update HCON.

Devices in SMIT in the SMIT Overview in *General Concepts and Procedures* is the section of SMIT used to configure the adapters used with HCON.

Configuring HCON

In order for HCON to establish a session with a host computer, it must have information about both the user running the session and the session itself. The configuration of HCON provides this information.

For HCON to have the necessary information about the user, the root user must register each user. When the user is registered, HCON stores the user's login ID in an Object Data Manager (ODM) database in the **/usr/lib/hcon** directory. In addition, HCON creates two ODM object classes: the **usrprofs** class and the **usrdflts** class, that are used to store the user's session profiles. The **usrprofs** class consists of the **$HOME/usrprofs** and **$HOME/usrprofs.vc** files. The **usrdflts** class consists of the **$HOME/usrdflts** and **$HOME/usrdflts.vc** files.

In addition to being registered, each HCON user must have at least one session profile defined before using HCON. The HCON programs consult the profile for information about the host and options to use. HCON users can define session profiles for themselves. In addition, the root user has the option of defining session profiles for any HCON user.

Registering HCON Users

You must be the root user to register an HCON user. The user you are registering must already be a valid AIX user. SMIT provides online help for the HCON administrative functions. These functions allow a user with root authority to add, remove, and list HCON users. In addition, using a shell procedure created by SMIT can be useful when you are defining the same users on more than one system. To start SMIT for use with HCON, run the **smit hcon** command.

Alternatively, you can add users at the command line with the **mkhconu** command, remove them with the **rmhconu** command, and list them with the **lshconu** command.

Defining HCON Session Profiles

If you are an HCON user, you can define session profiles for yourself. If you wish to define profiles for another user without logging in as that user, you must be the root user. Each user for whom you are creating profiles must have been registered as an HCON user.

SMIT provides HCON user functions. These functions allow you to add, change, remove, and list session profiles for the current user. A user with root authority can also work with profiles belonging to any HCON user. The SMIT dialogs provide online help and allow you to select session characteristics from lists of options rather than entering flags at the command line. In addition, using a shell procedure created by SMIT can be useful when you are defining multiple profiles. To start SMIT for use with HCON, run the **smit hcon** command.

Alternatively, you can work with session profiles from the command line. Use the **mkhcons** command to create a session profile or the **chhcons** command to change an existing profile. The **lshconp** command lists a user's session profiles, the **lshcons** command lists the characteristics of a session, and the **rmhcons** command removes a session profile.

For further information about configuring HCON, read the procedure How to Configure HCON on page 4–56.

Related Information

To start SMIT for use with HCON, use the **smit hcon** command. The **smit** command also discusses using the SMIT log file to create shell procedures.

The **mkhconu** command registers HCON users. The **rmhconu** command removes HCON users. The **lshconu** command lists HCON users.

The **mkhcons** command creates a session profile. The **chhcons** command changes an existing session profile. The **lshconp** command lists a user's session profiles. The **lshcons** command lists the characteristics of a profile. The **rmhcons** command removes a session profile. The **clhcons** command displays the session type of an existing profile.

HCON Host Configuration Overview on page 4–42 discusses System/370 host configuration requirements.

Understanding HCON Explicit and Implicit Logon in *Communications Programming Concepts* discusses setting up AUTOLOG and LAF profiles used to log in to a host system.

List of HCON Configuration Options

The System Management Interface Tree (SMIT) provides the following options for HCON configuration. To start SMIT for use with HCON, issue the **smit hcon** command.

Administrative Functions

These functions allow a user with root authority to add, remove, and list HCON users.

Add an HCON User

Adds a user to HCON and sets the user's defaults to match the system defaults. The user must be an AIX user and must not be already defined to HCON. You must have root authority to use this option.

Remove an HCON User

Removes a specified HCON user and all of that user's HCON session profiles. You must have root authority to use this option.

List HCON Users

Lists the login names of all users who have been defined to HCON. You must have root authority to use this option.

User Functions

These functions allow you to add, reconfigure, remove, and list session profiles for the current user. A user with root authority can work with profiles belonging to any HCON user.

Reconfigure a Session

Modifies an existing HCON session profile belonging to the current user. If you have root authority, you can also use this option to modify a profile belonging to any HCON user.

Reconfigure non-SNA DFT Display Session

Modifies the profile of a non-SNA DFT display session.

Reconfigure HIA Display Session

Modifies the profile of an HIA display session.

Reconfigure non-SNA DFT Printer Session

Modifies the profile of a non-SNA DFT printer session.

Remove a Session

Removes a session profile for the current user. If you have root authority, you can also use this option to remove a profile belonging to any HCON user.

Show Session Characteristics

Displays all the characteristics of a specified session profile belonging to the current user. If you have root authority, you can also use this option to display the characteristics of a profile belonging to any HCON user.

Add a Session

Creates an HCON session profile for the current user. If you have root authority, you can also use this option to create a profile for any HCON user.

List all Sessions

Lists the name and session use of all defined HCON session profiles belonging to the current user. If you have root authority, you can also use this option to list the profiles belonging to any HCON user.

Customizing HCON

Several aspects of the HCON system can be customized. They are:

- The colors and highlighting used on your terminal to display 3270 field and character attributes
- The keys you use to designate 3270 and HCON functions
- The names you use to define the keys, functions, colors, and attributes when customizing HCON
- The type of terminal you use with HCON.

Customizing Color and Keyboard Definition Tables

The HCON color and keyboard definition utilities can be used to customize the way 3270 field and character attributes are displayed on your terminal, and the location on the keyboard to which emulator keys are mapped.

Using color customization you can, for example:

- Change the background and foreground colors of the status line (on a color terminal).
- Set 3270 unprotected normal fields to blink (if blink is supported on your terminal).

Using keyboard customization you can, for example:

- Set a control key sequence on your system to delete a word when editing a host file.
- Set a key on your keyboard to have no function on the host.
- Set a key on your keyboard to produce a superscript or subscript on the host.
- Set a key on your keyboard to produce a character such as a ç (c cedilla) or ¨ (umlaut) on the host.

Someone with write access to the **/usr/lib/hcon/e789_ctbl** and **/usr/lib/hcon/e789_ktbl** files can customize these tables for the entire HCON system since definitions in these two files become the default for the system. Additional binary color and keyboard definition tables can also be stored in the **/usr/lib/hcon** directory (or another directory to which all HCON users have access) for use by all users on the system.

In addition, individual users can make their own customized tables to override system defaults. A user can have multiple customized tables, which can be named with any valid AIX file name and stored in any directory. The color and keyboard tables to be used for a particular session are determined by the profile for that session.

For further information, see the procedures How to Customize the HCON Keyboard Definition Table on page 4–60 and How to Customize the HCON Color Definition Table on page 4–59.

Changing Color, Key, and Attribute Names

HCON provides files listing the color, key, and attribute names that are used to customize the color and keyboard definition tables. These HCON files allow the root user to change the names of colors, attributes, keys, and functions to their equivalents in any language. Thereafter, HCON users must use the new names to define these colors, attributes, functions, and keys to HCON.

HCON provides three files, which are stored in the **/usr/lib/hcon** directory: the **func_names** file, the **keynames** file, and the **nls_names** file. The **keynames** and **func_names** files control the names used to designate key sequences and 3270 functions when customizing the HCON keyboard. The **nls_names** file controls the names used to designate colors and attributes when customizing HCON 3270 field display attributes. Edit the files with any ASCII editor to change the names.

Note: The names defined in the **func_names**, **keynames**, and **nls_names** files affect the entire color and keyboard customization process. If the names in these files are changed, all HCON users must use the new names as input to the **e789cdef** and **e789kdef** commands.

Defining a New Terminal

When installed, HCON supports all RISC System/6000 high function terminals (HFTs) and the following ASCII terminals: IBM 3151, IBM 3161, IBM 3162, IBM 3163, DEC VT100, DEC VT220, and WYSE WY–50 terminals. If you wish to use another terminal with HCON on your system, you must define the new terminal's key values for HCON. How to Define a New Terminal for HCON on page 4–61 discusses the steps involved in defining a new terminal.

Related Information

How to Customize the HCON Keyboard Definition Table on page 4–60 and How to Customize the HCON Color Definition Table on page 4–59 discuss the steps needed to define new tables for HCON.

Use the **func_names** file format to customize HCON function names, the **keynames** file format to customize key names, and the **nls_names** file format to customize color and attribute names.

How to Define a New Terminal for HCON on page 4–61 discusses the steps involved in defining a new terminal.

Maintaining HCON

The main tasks involved in maintaining HCON are:

- Updating the configuration and customization
- Problem solving.

As users on your system change, you will need to add and delete users from HCON. Additionally, you may need to create new session profiles for existing users. For example, if a new 3270 device were added to your system, you might create non–SNA DFT profiles for users who need to use the new device.

Problem solving for HCON involves several areas. If users are experiencing difficulties, you can check their session profiles to make sure all of the parameters are correctly defined. In addition, you may need to recover from interrupted file transfers or free up system resources if a session is interrupted. You may need to use the **ps** command to check on HCON processes. Finally, if you are experiencing problems with host connections, check the hardware devices connecting you to the host, and check the host configuration.

You can use the **ps** command to check on HCON processes. The **hcondmn** subsystem should be running at all times. If any sessions are running explicitly, there should be an **e789** process in the process display, and for each individual session, there should be one **e789x** process. If a file transfer is in progress, the **dfxfer** process should also be running. If an API program or file transfer is running implicitly, there should be an associated **e789x** process in the process display.

Recovering from Interrupted HCON File Transfers

File transfers can be interrupted by the operator before completion. In addition, a file transfer can be interrupted by any unrecoverable communication error. When a transfer is interrupted, the state of the transfer is saved in a RESTART file in the user's **$HOME** directory. An interrupted file transfer can thus be restarted without loss of data. For an explicit file transfer, the restart file is the **$HOME/x_fxfer.r** file. For an implicit file transfer, the restart file is the **$HOME/i_fxfer.r** file. A maximum of five file transfer requests can be saved in RESTART files.

Automatic Recovery

If the host communication is lost or disconnected, an implicit file transfer may attempt to recover by reconnecting and logging back on to the host. The length of time that the **dfxfer** process will attempt recovery is determined from the File Transfer Recovery Time value in the session profile. Once the host communication is reestablished, the **dfxfer** process resumes the file transfer from the beginning. If communication cannot be reestablished, the file transfer program generates a RESTART file.

Restarting a File Transfer Using the fxfer –R Flag

If a file transfer has been interrupted, and the user runs a new file transfer, the **fxfer** command reminds the user that a RESTART file has been created. The RESTART menu gives the user the choice of restarting the interrupted file transfer, deleting the RESTART file, exiting from the file transfer program, or continuing with the current file transfer. If the user elects to continue the present file transfer, the RESTART file for the previous transfer can be deferred or deleted.

The user can also use the **–R** flag of the **fxfer** command to deliberately restart an interrupted file transfer. The **–R** flag brings up a menu from which the user can choose whether to restart the file transfer, delete the RESTART file, or exit.

If the user chooses to restart a file transfer, the **fxfer** command restarts the previous file transfer using the information saved in one of the RESTART files, the **$HOME/x_fxfer.r** file or the **$HOME/i_fxfer.r** file. The **$HOME/i_fxfer.r** file contains RESTART information for implicit logon queues. The **$HOME/x_fxfer.r** file contains RESTART information for explicit logon queues.

Working with Interrupted HCON Sessions

HCON sessions can be interrupted by situations such as a power failure or a loss of the connection with the host. If a session is interrupted, the processes associated with it may not have been stopped. These processes must be stopped with the **e789cln** command, which stops specified HCON sessions and processes associated with those sessions.

The **e789cln** command can be used by any HCON user to stop that user's own sessions. The command can also be used by someone with root user authority to stop the sessions of another user. The command can be used to stop one session, multiple sessions, or all sessions for a given user.

The **e789cln** command should be used only when there is a problem requiring that the processes associated with a session be stopped. Using the **e789cln** command to stop a session that is proceeding normally can cause errors. Use the QUIT emulator key (by default, the Ctrl-D key combination) to stop a normal session.

Troubleshooting HCON Problems Related to Host Configuration

The following problems that may occur with HCON relate to the manner in which the host computer system and the control unit are configured:

- *The file transfer always times out on a VM/VTAM host.* This can occur if the PSERVIC field of the MODEENT macro has not been updated for HCON and VTAM has not been restarted after changing the PSERVIC value.
- *The host logo does not appear on the emulator screen.* It is sometimes necessary to press the ENTER key to make the logo appear. If pressing the ENTER key does not work, check for one of the following conditions:
  - The coaxial cable is unplugged.
  - The control unit (3174, 3274, or 5088) is not operational.
  - The link speed between the 5088 control unit and the RISC System/6000 is incorrectly configured.
  - The control unit channel address on the 5088 is improperly set.
  - The host channel address is configured for a device model number not supported by HCON.
  - The control unit is not configured correctly.
  - The network or terminal address for the session has not been configured correctly on the host.
- *The API or the file transfer times out when messages longer than 4096 bytes* are sent to the host computer. This can occur on VM and MVS VTAM host computer systems that have VTAM IOBUF parameters optimized for terminal traffic.

To fix these problems, consult the section of the HCON Host Configuration Overview for the type of host or control unit involved.

The following problem that may occur relates to how the host configuration is reflected in the session profile:

- *Program checks during file transfer while running an API/API application.* This problem can occur using **pvm** or **passthru** facility on the System/370 host if the Maximum I/O Buffer Size is greater than 3456 bytes. Check the Maximum I/O Buffer Size in the session profile.

For further information about processes affected by the **e789cln** command, read Understanding HCON Processes on page 4–29. For further information about file transfers, read Understanding the HCON File Transfer Process on page 4–31. For further information about host configuration, read the HCON Host Configuration Overview on page 4–42.

Related Information

The **fxfer** command starts and restarts file transfers.

The **ps** command can be used to check on HCON processes.

HCON Host Configuration Overview

Before users on your system can use HCON to communicate with an IBM System/370 host computer, the host system must meet certain configuration requirements. Different systems have differing requirements. The following pages discuss requirements for four different host system types:

- MVS/TSO Host Configuration Requirements
- VM/CMS Host Configuration Requirements
- MVS and VM/VTAM Version 3 Host Configuration Requirements
- Host Terminal Control Unit Configuration Requirements.

Once a host is properly configured for use with HCON, each HCON user must be provided with an appropriate login ID on the host computer. Also, before programs written with the HCON API can be used to communicate with a host system, the host API must be installed on that system.

For further information, read Configuring an MVS/TSO Host for HCON on page 4–42, Configuring a VM/CMS Host for HCON on page 4–43, Configuring MVS and VM/VTAM Version 3 Hosts for HCON on page 4–44, or Configuring a Host Terminal Control Unit for HCON on page 4–47, depending on the type of host computer to which you will be connecting.

For information about debugging host connections, read Troubleshooting HCON Problems Related to Host Configuration on page 4–41.

Related Information

Installing and Updating HCON on page 4–33 and Configuring HCON on page 4–35.

How to Install the HCON MVS/TSO Host API and How to Install the HCON VM/CMS Host API in *Communications Programming Concepts*.

Configuring an MVS/TSO Host for HCON

Before you can use HCON to connect to an MVS/TSO host, the MVS I/O system generation requires that the RISC System/6000 running the HCON emulator, the HCON file transfer program, or the HCON API be defined to MVS as one or more 3278 display terminals.

Note: 3081 MVS installations may specify a CNTLUNIT macro for the 5088 control unit to which the RISC System/6000 is attached. The CNTLUNIT macro is ignored by the MVS I/O system generation. However, it must be specified for the I/O Configuration Program (IOCP) that must be run for the Model 3081 processor.

The RISC System/6000 must be defined to MVS in the IODEVICE macro as a 3278 Display Device. The IODEVICE parameters for an HCON terminal defined as a 3278 terminal are the same as those for a regular 3278 terminal.

The following is a sample IODEVICE definition for HCON. The model information is ignored.

```
DEVRT780 IODEVICE UNIT=3278,ADDRESS=780,
                  MODEL=2,
                  FEATURE=(EBKY3277,KB78KEY,AUDALRM,
                  DOCHAR,SELPEN,PTREAD),OFFLINE=NO
```

Note: Lines ending with a comma are continued in column 72.

Related Information

Installing and Updating HCON on page 4–33 and Configuring HCON on page 4–35 discuss setting up HCON on the RISC System/6000.

How to Install the HCON MVS/TSO Host API in *Communications Programming Concepts.*

OS/VS2 System Programming Library: System Generation Reference and *MVS/XA Installation System Generation* provide more information about MVS/TSO host configuration.

Configuring a VM/CMS Host for HCON

The VM nucleus DMKRIO I/O definition requires the following for use with HCON:

- The RISC System/6000 running the HCON emulator, the HCON file transfer program, or the HCON API is defined to VM as one or more 3278 display terminals. If 3286/87 printer emulation is desired, one or more 3286 definitions are required.
- The 5088 control unit that is attached to the RISC System/6000 is defined to VM as a 3274. When the RISC System/6000 is attached to a 5088 that is defined to VM in this manner, the 5088 is running in 3274 emulation mode.

The RISC System/6000 must be defined to VM in the RDEVICE macro as a 3278 Display Device or a 3286 printer device. The RDEVICE parameters for a RISC System/6000 defined as a 3278 terminal or 3286 printer are the same as those for a regular 3278 display terminal or 3286 printer, respectively.

The following is a sample RDEVICE definition for HCON. The model information is ignored.

```
DEVRT780 RDEVICE DEVTYPE=3278,ADDRESS=780,
                 MODEL=2
DEVRT781 RDEVICE DEVTYPE=3286,ADDRESS=781,
                 MODEL=2
```

Note: Lines ending with a comma are continued in column 72.

The 3274 or 5088 must be defined to VM in the RCTLUNIT macro as a 3274 control unit.

The RISC System/6000 must be attached to VM using either a 3274 or 5088 control unit. If the RISC System/6000 is attached to a 3274 control unit, the 3274 is defined to VM as a 3274. If the RISC System/6000 is attached to a 5088 control unit, the 5088 is defined to VM as a 3274 and runs in 3274 emulation mode.

Following is a sample RCTLUNIT definition of a 16-port 5088 Model 1:

```
CTLRT770 RCTLUNIT ADDRESS=770,UNIT=3274,
                  FEATURE=16–DEVICE
```

Following is a sample RCTLUNIT definition of a 32-port 3274 control unit:

```
CTLRT770 RCTLUNIT ADDRESS=770,UNIT=3274,
                  FEATURE=32–DEVICE
```

Note: Lines ending with a comma are continued in column 72.

Related Information

Installing and Updating HCON on page 4–33 and Configuring HCON on page 4–35 discuss setting up HCON on the RISC System/6000.

How to Install the HCON VM/CMS Host API in *Communications Programming Concepts* discusses installing the HCON API on the host.

Virtual Machine/System Product Planning Guide and Reference and *Virtual Machine/System Product High Performance Option Planning Guide and Reference* provide more information about VM/CMS host configuration.

Configuring MVS and VM/VTAM Version 3 Hosts for HCON

MVS Version 3 VTAM hosts require the following for use with HCON:

- The RISC System/6000 must be defined as one or more VTAM terminals. The definition must be made using a LOCAL statement in a VTAM node defined by a member of SYS1.VTAMLST.
- The MODETAB operand of the LOCAL statement must point to a VTAM logon mode table member of SYS1.VTAMLIB, with an applicable logon mode name defined by a MODEENT macro, with bit 8 of the PSERVIC operand of the MODEENT macro set to 1.

A VM Version 3 VM/VTAM host requires the following for use with HCON:

- The RISC System/6000 must be defined as one or more VTAM terminals. This definition must be made using a LOCAL statement in a VTAM node defined by a CMS file, with a file name of the node name and a file type of VTAMLST.
- The MODETAB operand of the LOCAL statement must point to a VTAM logon mode table member of VTAMUSER LOADLIB, with an applicable logon mode name defined by a MODEENT macro, with bit 8 of the PSERVIC operand of the MODEENT macro set to 1.
- The **bufsize** parameter for the VTAM IOBUF storage pool must be increased from 64 bytes to at least 256, 285, or 286 bytes, because 64 bytes is too small a **bufsize** for messages of 32K bytes.

Setting Up the LOCAL Statement to Define the RISC System/6000 As a VTAM Terminal

The RISC System/6000 must be defined as one or more VTAM terminals in a local non-SNA major node. This is done by coding a LOCAL statement for the RISC System/6000 in the member of SYS1.VTAMLST for MVS or in the CMS file with a file type of VTAMLST for VM corresponding to the macro mode. The parameters for the LOCAL statement for a RISC System/6000 are the same as those for a 32xx display terminal or 3286 printer.

The following is a sample definition of a local non-SNA major node containing LOCAL statement definitions for two nodes to be used by a RISC System/6000 that is using HCON:

```
        LBUILD
DEVRT780 LOCAL TERM=3277,CUADDR=780,
               ISTATUS=ACTIVE,FEATUR2=(EDATS,ANKEY,
               MODEL2,PFK,SELPEN),LOGTAB=INTTAB,
               MODETAB=RSTINCLM,DLOGMOD=MOD2,
               USSTAB=USST327X,LOGAPPL=NETMON

DEVRT781 LOCAL TERM=3286,CUADDR=781,
               ISTATUS=ACTIVE,FEATUR2=(EDATS,ANKEY,
               MODEL2,PFK,SELPEN),LOGTAB=INTTAB,
               MODETAB=RSTINCLM,DLOGMOD=MOD2,
               USSTAB=USST327X,LOGAPPL=NETMON
```

Note: Lines ending with a comma are continued in column 72.

Setting Up the MODEENT Macro and the PSERVIC Parameter in the LOGMODE Table

The LOCAL statement in the local non-SNA major VTAM node that defines the RISC System/6000 contains a MODETAB parameter. This MODETAB parameter must be the name of a VTAM logon mode table, and either be a member of the SYS1.VTAMLIB for MVS or a member of VTAMUSER LOADLIB for VM/CMS. This logon mode table consists of one or more MODEENT macros, each for a different logon mode name specified by the LOGMODE parameter, and each specifying a different set of session parameters. For each logon mode name/MODEENT macro applicable to the RISC System/6000, the PSERVIC parameter must have bit 8 set properly for the session type.

For display sessions, bit 8 of the 24-hex-digit PSERVIC parameter of the MODEENT macro associated with the RISC System/6000 running HCON must be set to 1. For example, for display sessions the parameter must be set as follows:

```
MODEENT ...,PSERVIC=X'00 80 00 00 00 00 00 00 00 00 03 00'
```

In addition, bytes 6–10 should be set as shown so that VTAM will query HCON for the alternate screen size.

For printer sessions, the PSERVIC parameter must be set as follows:

```
MODEENT ...,PSERVIC=X'00 80 00 00 00 00 00 00 00 00 02 00'
```

The IBM-supplied default logon mode table is named INSTINCLM. The following is an example of a replacement logon mode table named RSTINCLM with the required PSERVIC parameter setting for 3278/79 emulation:

```
RSTINCLM MODETAB ,
*
* 3270 NON-SNA WITH EXTENDED DATA STREAM:
*

MOD2     MODEENT LOGMODE=MOD2,FMPROF=X'02',
                 TSPROF=X'02',PRIPROT=X'71',
                 SECPROT=X'40',COMPROT=X'2000',
                 RUSIZES=X'0000',
                 PSERVIC=X'00 80 00 00 00 00 00 00 00 00 03 00'
*
                 MODEEND
                 END    ,
```

Note: Lines ending with a comma are continued in column 72.

Setting the bufsize Parameter

For VM/VTAM, the **bufsize** parameter for the VTAM IOBUF storage pool must be increased from the IBM default of 64 bytes to at least 256 bytes in order to run file transfer operations and API applications efficiently. The IOBUF **bufsize** is the size in bytes of each buffer in the IOBUF storage pool.

HCON includes an API sample program, called **g32_sampl**, that transfers one megabyte of memory to and from the host in 1K, 2K, 4K, 8K, 16K, and 32K byte messages. The HCON user can also change the HCON message size to accommodate the host configuration. The Maximum I/O buffer size profile characteristic specifies the maximum message size allowed by the host for both file transfer and API applications.

Whether or not a workstation is attached, the presence of SDLC or BSC lines coming into VTAM through a communications adapter places an additional requirement on the **bufsize** parameter, as follows:

- If VTAM has SDLC lines attached through a communications adapter, the **bufsize** value must be at least 196 bytes.
- If VTAM has BSC lines attached through a communications adapter, the **bufsize** parameter must be at least 285 bytes.
- If VTAM has BSC lines attached through a communications adapter and the network also includes channel-attached SNA devices, the **bufsize** value must be at least 286 bytes.

The following table summarizes the minimum values required for VM/VTAM IOBUF **bufsize** for the emulating RISC System/6000, with and without communications adapters:

Minimum I/O Buffer Size

Size	Situation
256	No SDLC or BSC lines through communication adapter
256	SDLC lines through communications adapter
285	BSC lines through communications adapter
286	BSC lines through communications adapter and channel-attached SNA devices.

The **bufsize** parameter for the IOBUF storage pool is set when VTAM is started. VTAM then reads a list of user-specified start options from CMS file ATCSTR00 VTAMLST. However, the operator can specify a supplemental list of start options with the LIST=yy operand of the VTAM command.

To change the **bufsize** parameter, a supplemental list of start options should be created as, for example, CMS file ATCSTRRT VTAMLST. This member should contain an IOBUF statement in the following format:

IOBUF=(,*BufferSize*,,,,)

where the *BufferSize* variable defines the value of the **bufsize** parameter. The **bufsize** value must be at least 256, 285, or 286 bytes as defined in the preceding table. For example,

```
IOBUF=(,256,,,,)
```

Then, when VTAM is started with the LIST=yy operand pointing to the new start options file, as in:

```
VTAM START LIST=RT
```

each buffer in the IOBUF storage pool is the newly specified **bufsize** length in bytes.

Related Information

Installing and Updating HCON on page 4–33 and Configuring HCON on page 4–35 discuss setting up HCON on the RISC System/6000.

How to Install the HCON VM/CMS Host API in *Communications Programming Concepts.*

VTAM Version 3 Release 3 Installation and Resource Definition and *VTAM Version 3 Release 3 Customization* provide further information about VTAM configuration.

Configuring a Host Terminal Control Unit for HCON

HCON is supported only on the following non-SNA DFT host control units. Each has different configuration requirements:

- 3274 control unit
- 3174 control unit
- 5088 control unit.

3274 Control Unit

Following are the three 3274 controller customization requirements (for 3274 Configuration Support D) for the RISC System/6000 running the HCON 3278/79 emulator, the 3270 file transfer program, or the API:

- The 3274 must be customized to include Distributed Function Terminal (DFT) support.
- The controller must be customized for Extended Data Stream processing.
- If a RISC System/6000 is to support multiple sessions, the controller must be customized for Multiple Interactive Screens.

3174 Control Unit

The 3174 Control Unit must be customized for multiple logical terminal (MLT) support if HCON on the RISC System/6000 is to support multiple sessions.

5088 Control Unit

Following are the two 5088 controller requirements for the RISC System/6000 running the HCON emulator, the 3270 file transfer program, or the API:

- The maximum and minimum control unit channel addresses of RISC System/6000 machines attached to the 5088 Model 1 or Model 2 must be set into the 5088 DIP switches, and the HIA **lbond** and **ubond** parameters must fall within the range of 5088 DIP switch values.
- Local coax speed of the RISC System/6000 to 5088 coaxial cable attachment must be specified the same in the HIA **lsped** parameter and in switch U26–6 of the 5088–1 or 5088–2.

Note: A host MVS or VM address of, for example, 77F results in a control unit channel address of 7F.

Control Unit Channel Addresses

The maximum control unit channel address is set as an 8-bit binary value in the 5088 DIP switches U48–8 through U48–1 (U48–8 is the most significant bit and U48–1 is the least significant bit).

For example, a maximum MVS or VM address of 77F on the 5088 is set as:

Switch	Value
U48–8	0
U48–7	1
U48–6	1
U48–5	1
U48–4	1
U48–3	1
U48–2	1
U48–1	1

The value to set for the minimum RISC System/6000 control unit channel address is obtained as a coded value from a table in *IBM 5088 Models 1, 2, and 1R Graphics Controller Maintenance Information*. The coded value is then set into the DIP switches U33–4 through U33–1 of the 5088.

For example, the coded value for a minimum MVS or VM address of 770 on the 5088 corresponding to the maximum address of 77F is 1100 and is set as:

Switch	Value
U33–4	1
U33–3	1
U33–2	0
U33–1	0

Local Coax Speed

The **lsped** parameter and the DIP switch setting U26–6 of the 5088 Model 1 or 2 both refer to the speed of the coaxial cable link between the RISC System/6000 and the 5088 control unit. Once DIP switch U26–6 has been set on the 5088–1 or 5088–2, the **lsped** parameter for all the RISC System/6000 systems with a coaxial cable attachment to that 5088 must agree with switch U26–6.

The table below shows the meaning of the **lsped** parameter and U26–6 values, and the acceptable pairs of **lsped**/U26–6 values for the two possible local coax speeds:

Values for the lsped parameter		
Local Coax Speed	**lsped Value**	**Switch U26–6 Value**
1 megabit/second	1	1
2 megabits/second	2	0

Note: The HIA **lsped**, **lbond**, and **ubond** parameters are set during configuration of the System/370 Host Interface Adapter (HIA). To change these parameters, reconfigure the HIA using SMIT or the **chdev** command.

Related Information

Installing and Updating HCON on page 4–33 and Configuring HCON on page 4–35 discuss setting up HCON on the RISC System/6000.

Use Defining and Configuring Your System with SMIT in *General Concepts and Procedures* or the **chdev** command to reconfigure an System/370 Host Interface Adapter.

IBM 3270 Information Display System 3274 Control Unit Customizing Guide Configuration Support D, *3270 Information Display System, 3174 Subsystem Control Unit Customization Guide*, and *IBM 5088 Models 1, 2, and 1R Graphics Controller Maintenance Information* provide more information about Terminal Control Units.

List of HCON System Management Information

The information about managing the AIX 3270 Host Connection Program/6000 (HCON) falls into the following general categories:

Commands are listed beginning on page 4–50.

Files and File Formats are listed on page 4–52.

Related Information

The HCON Overview on page 4–1 introduces HCON. The HCON Overview for Programming in *Communications Programming Concepts* discusses the HCON Application Programming Interface.

List of HCON System Management Commands

The HCON commands fall into several categories. Some commands fall into more than one category.

Commands for setting up HCON

Commands for working with HCON users

Commands for working with session profiles

Commands for customizing the color and keyboard definition tables

Commands for working with HCON sessions

Alphabetical list of commands.

Commands for Setting Up HCON

mkhcons command Creates a session profile.

mkhconu command Registers HCON users.

sthcondmn command Starts the **hcondmn** subsystem

Commands for Working with HCON Users

mkhconu command Registers HCON users.

lshconu command Lists HCON users.

rmhconu command Removes an HCON user.

Commands for Working with Session Profiles

chhcons command Changes a session profile.

clhcons command Returns the type of a session profile.

mkhcons command Creates a session profile.

lshconp command Lists a user's session profiles.

lshcons command Lists the characteristics of a session.

rmhcons command Removes a session profile.

Commands for Customizing the Color and Keyboard Definition Tables

e789cdef command Creates a binary color definition table.

e789kdef command Creates a binary keyboard definition table.

Commands for Working with HCON Sessions

e789 command	Starts one or more specified HCON sessions.
e789cln command	Removes IPC resources and HCON processes left over from one or more abnormally terminated sessions.
fxfer command	Transfers files between the RISC System/6000 and the host.

Alphabetical List of HCON Management Commands

chhcons command	Changes a session profile.
clhcons command	Lists the type of a session profile.
e789 command	Starts one or more specified HCON sessions.
e789cdef command	Creates a binary color definition table.
e789cln command	Removes IPC resources and HCON processes left over from one or more abnormally terminated sessions.
e789kdef command	Creates a binary keyboard definition table.
fxfer command	Transfers files between the RISC System/6000 and the host.
mkhcons command	Creates a session profile.
mkhconu command	Registers HCON users.
lshconp command	Lists a user's session profiles.
lshcons command	Lists the characteristics of a session.
lshconu command	Lists HCON users.
rmhcons command	Removes a session profile.
rmhconu command	Removes an HCON user.
sthcondmn command	Starts the **hcondmn** subsystem

Related Information

See the *Commands Reference* for more information about the HCON commands.

Understanding HCON Users on page 4–20, Understanding HCON Sessions on page 4–20, Understanding HCON Session Profiles on page 4–21, and Configuring HCON on page 4–35 discuss uses of the user and session profile commands.

Customizing HCON on page 4–37 explains how the **e789cdef** and **e789kdef** commands are used.

Maintaining HCON on page 4–39 discusses some of the uses of the **e789cln** and **fxfer** commands.

List of HCON System Management Files and File Formats

Files

The HCON files are located in the **/usr/lib/hcon** directory:

e789_ctbl Default binary color definition table

e789_ktbl Default binary keyboard definition table.

File Formats

The HCON file formats are used to customize HCON and are located in the **/usr/lib/hcon** directory:

e789_ctbl.p Source for the default binary color definition table

e789_ktbl.p Source for the default binary keyboard definition table

func_names Keyboard function names

keynames Key names

nls_names Color and attribute names.

Related Information

See the *Files Reference* for more information about the HCON files and file formats.

The **e789cdef** command, **e789kdef** command.

Customizing HCON on page 4–37 explains how the HCON files and file formats and the **e789cdef** and **e789kdef** commands are used.

How to Install the AIX 3270 Host Connection Program/6000 (HCON) Licensed Program

You can install the AIX 3270 Host Connection Program/6000 licensed program in the following ways:

- From preloaded disk
- From tape
- From a network server
- From diskette.

The following sections describe how the licensed program is packaged, the system requirements for installation, and procedures for installing the licensed program.

How the AIX 3270 Host Connection Program Licensed Program is Packaged

AIX 3270 Host Connection Program licensed program consists of the following parts:

hcon.obj	Contains the AIX 3270 Host Connection Program/6000 program.
hconm*Language*.**msg**	Contains the messages and helps in the specified *Language* for the run time environment. If you install multiple languages for a product, be sure that you install the preferred (or primary) language first.

The licensed program may also contain update files.

Prerequisite Tasks or Conditions

- You must log in as the root user.
- You must have installed the AIX Base Operating System (BOS) Runtime (part of the IBM AIX Base Operating System licensed program).
- You must have installed the Base Application Development Toolkit (part of the AIX Base Operating System licensed program) before you attempt to statically bind any AIX 3270 Host Connection Program modules.
- The **hcondmn** subsystem must not be active. To determine if the subsystem is active, enter:

  ```
  /bin/lssrc -s hcon
  ```

  If the subsystem is active, enter the following to terminate it:

  ```
  /bin/stopsrc -f -s hcon
  ```

  Warning: Be sure all HCON activity has ceased before stopping the subsystem.

Notes:

1. To *run* HCON, you must install and configure the appropriate communications devices. However, you can *install*, *configure*, and *customize* HCON either before or after installing and configuring the devices.
2. When you have successfully completed the HCON installation, you must configure HCON before it can be used on your system.
3. If you are performing an initial installation using the *Installation Instructions for AIX*, use those procedures to continue.

Procedure to Install for Preloaded Disk

1. Enter the following on an AIX command line:

   ```
   smit instupdt
   ```

   This command invokes the System Management Interface Tool (SMIT), which presents a menu driven environment for the installation process.

2. Follow the directions and answer the prompts in the SMIT Install menus. Select **hcon.obj**, **hconm***Language*.**msg**, and any updates for the licensed program.

   Messages display as each part successfully completes installing.

3. Start the **hcondmn** subsystem. To start the subsystem, enter:

   ```
   /usr/lib/hcon/sthcondmn
   ```

Procedure to Install from Tape

1. Insert the tape containing **hcon.obj** into the tape drive.
2. Enter the following on an AIX command line:

   ```
   smit instupdt
   ```

   This command invokes the System Management Interface Tool (SMIT), which presents a menu driven environment for the installation process.

3. Follow the directions and answer the prompts in the SMIT Install menus. Select **hcon.obj**, **hconm***Language*.**msg**, and any updates for the licensed program.

   Messages display as each part successfully completes installing.

4. Start the **hcondmn** subsystem. To start the subsystem, enter:

   ```
   /usr/lib/hcon/sthcondmn
   ```

Procedure to Install over a Network

1. Refer to How to Install over a Network in the *Installation Kit for IBM AIX Version 3 for RISC System/6000* for instructions on how to set up the server and how to download files to the client. In addition, if you will be mounting HCON software from a server onto a client system, read How to Set Up HCON in a Codeserver Environment on page 4–58.
2. Select Optional Program Products from the System Startup menu.
3. Follow the directions and answer the prompts in the SMIT Install menus. Select the files that contain **hcon.obj**, **hconm***Language*.**msg**, and any updates for the AIX 3270 Host Connection Program/6000.

   Messages display as each part successfully completes installing.
4. Start the **hcondmn** subsystem. To start the subsystem, enter:

   ```
   /usr/lib/hcon/sthcondmn
   ```

Procedure to Install from Diskette

1. Insert the first **hcon.obj** diskette into the diskette drive.
2. Enter the following on an AIX command line:

   ```
   smit instupdt
   ```

   This command invokes the System Management Interface Tool (SMIT), which presents a menu driven environment for the installation process.
3. Follow the directions and answer the prompts in the SMIT Install menus. First install **hcon.obj**, then install **hconm***Language*.**msg**, and finally apply the updates.

   Messages display as each part successfully completes installing.
4. Start the **hcondmn** subsystem. To start the subsystem, enter:

   ```
   /usr/lib/hcon/sthcondmn
   ```

Related Information

The **installp** command, **updatep** command, **instupdt** command, **lssrc** command, **stopsrc** command, **smit** command, and **sthcondmn** command. For information about these commands, use the InfoExplorer information retrieval facility. This information is also included in the *AIX Commands Reference for IBM RISC System/6000.*

Installation Instructions for IBM AIX Version 3 for RISC System/6000.

Installing and Updating HCON on page 4–33.

How to Set Up HCON in a Codeserver Environment on page 4–58, How to Configure HCON on page 4–56.

The System Management Interface Tool (SMIT) Overview in *General Concepts and Procedures.*

How to Configure HCON

Prerequisite Tasks or Conditions

- The HCON program and message index must be installed on your system.
- To register HCON users, you must have root user authority.
- To create an HCON session profile for another user, you must have root user authority.

Note: To *run* HCON, you must install and configure the appropriate communications devices. However, you can *install* and *configure* HCON either before or after installing and configuring the devices.

Procedure

1. Register HCON users using the **smit hcon** command or the **mkhconu** command.
2. Define HCON session profiles using the **smit hcon** command or the **mkhcons** command.
3. Configure the host for use with HCON.
4. Make sure that each HCON user has an appropriate login ID on the host.

Optional Additional Steps

Optionally, you can:

- Install the HCON API on a System/370 host system. This is necessary if HCON users will be running HCON API programs.
- Customize the keyboard and color definition tables for HCON (either for the entire system or an individual user).
- Develop shell procedures that invoke the HCON file transfer process.
- Set up implicit logon procedures for connecting to the host computer, using AUTOLOG or LAF.
- Add a new terminal for use with HCON.

Related Information

How to Install HCON on page 4–53 explains the procedures for installing the HCON software.

The **smit** command starts SMIT. See the **smit** command for information about using SMIT output. The **smit hcon** fastpath starts SMIT for use with HCON. Devices in SMIT in the SMIT Overview in *General Concepts and Procedures* discusses using SMIT to configure communications adapters.

HCON Host Configuration Overview on page 4–42 discusses host configuration requirements for HCON.

How to Customize the HCON Keyboard Table on page 4–60, How to Customize the HCON Color Table on page 4–59, How to Create a Shell Script for HCON File Transfers on page 4–63, and How to Define a New Terminal for HCON on page 4–61 explain procedures for customizing HCON.

Understanding HCON Explicit and Implicit Logon, How to Use an AUTOLOG Profile, and How to Use a Logon Assist Feature Script in *Communications Programming Concepts* discuss implicit login procedures.

Understanding the Host Interface for HCON, How to Install the HCON MVS/TSO Host API, and How to Install the HCON VM/CMS Host API in *Communications Programming Concepts* discuss installing the HCON Host API.

The **mkhconu** command and **mkhcons** command are used in HCON configuration. The **lshconu** command can be used to list HCON users.

How to Set Up HCON in a Codeserver Environment

Prerequisite Conditions

- You must have root user authority on both the codeserver and the client to carry out the steps in this procedure.
- The ODM database and the System Resource Controller (SRC) must be installed on the client system.

Procedure

1. Install the HCON software on the codeserver.
2. Connect the client system from which HCON will be run to an IBM System/370 host computer.
3. Log in as the root user on the client system.
4. Mount the **/usr/lib** and **/usr/bin** directories from the codeserver onto the client.

   Note: HCON cannot be run on the client unless these directories are mounted from the server.
5. Run the **sthcondmn** command on the client. The **sthcondmn** command starts the **hcondmn** subsystem and places instructions for the SRC in the ODM database so that the **hcondmn** subsystem is started each time you restart the client system. To run the command, enter the following on the client system:

   ```
   /usr/lib/hcon/sthcondmn
   ```
6. Configure HCON on the client system.

Related Information

The **sthcondmn** command and the **mount** command.

How to Install HCON on page 4–53, How to Configure HCON on page 4–56.

Installing and Updating HCON on page 4–33, Configuring HCON on page 4–35.

How to Customize the HCON Color Definition Table

Prerequisite Tasks or Conditions

- The HCON program and message index must be installed on your system.
- You must be registered as an HCON user to use the customized table.
- To customize the table for the entire system, you must have write access to the **/usr/lib/hcon** directory and the **/usr/lib/hcon/e789_ctbl** file.

Note: To *run* HCON, you must install and configure the appropriate communications devices. However, you can *install*, *configure*, and *customize* HCON either before or after installing and configuring the devices.

Procedure

A default color definition table is included in the HCON program. This table can be customized for the entire system, or a new table can be created for an individual user.

Customizing for the Entire System

To customize the color definition table for the entire system:

1. Save a copy of the original default color definition table by copying the **/usr/lib/hcon/e789_ctbl** file to another file.
2. Make a copy of the **/usr/lib/hcon/e789_ctbl.p** file.
3. Develop a customized color definition table source file by editing the copy of the **e789_ctbl.p** file. For instructions, read the **e789_ctbl.p** file format.
4. Create the new color definition table using the **e789cdef** command. Use the **–o** flag to name the new table. Name it **/usr/lib/hcon/e789_ctbl** so that it will become the default color definition table for the entire system. Specify the customized color definition table source file as the input file.

Customizing for an Individual User

To create a customized color definition table for yourself:

1. Copy the **/usr/lib/hcon/e789_ctbl.p** file to your **$HOME** directory or another directory to which you have write access.
2. Develop a customized color definition table source file by editing your copy of the **e789_ctbl.p** file. For instructions, read the **e789_ctbl.p** file format.
3. Create a new color definition table using the **e789cdef** command. Use the **–o** flag to name the new table. Specify the customized color definition table source file as the input file.

Note: To use the customized table, you must specify the name of the new table as the color definition table in your HCON session profiles.

Related Information

The **e789cdef** command and **e789_ctbl.p** file format.

The **/usr/lib/hcon/e789_ctbl** file.

How to Customize the HCON Keyboard Definition Table on page 4–60.

How to Customize the HCON Keyboard Definition Table

Prerequisite Tasks or Conditions

- The HCON program and message index must be installed on your system.
- You must be registered as an HCON user to use the customized table.
- To customize the table for the entire system, you must have write access to the **/usr/lib/hcon** directory and the **/usr/lib/hcon/e789_ktbl** file.

Note: To *run* HCON, you must install and configure the appropriate communications devices. However, you can *install*, *configure*, and *customize* HCON either before or after installing and configuring the devices.

Procedure

A default keyboard definition table is included with the HCON program. This table can be customized for the entire system, or a new table can be created for an individual user.

Customizing for the Entire System

To customize the keyboard definition table for the entire system:

1. Save a copy of the original default keyboard definition table by copying the **/usr/lib/hcon/e789_ktbl** file to another file.
2. Make a copy of the **/usr/lib/hcon/e789_ktbl.p** file.
3. Develop a customized keyboard definition table source file by editing the copy of the **e789_ktbl.p** file. For instructions, read the **e789_ktbl.p** file format.
4. Create the new keyboard definition table using the **e789kdef** command. Use the **–o** flag to name the new table. Name it **/usr/lib/hcon/e789_ktbl** so that it will become the default keyboard definition table for the entire system. Specify the customized keyboard definition table source file as the input file.

Customizing for an Individual User

To create a customized keyboard definition table for yourself:

1. Copy the **/usr/lib/hcon/e789_ktbl.p** file to your **$HOME** directory or another directory to which you have write access.
2. Develop a customized keyboard definition table source file by editing your copy of the **e789_ktbl.p** file. For instructions, read the **e789_ktbl.p** file format.
3. Create a new keyboard definition table using the **e789kdef** command. Use the **–o** flag to name the new table, and specify the customized keyboard definition table source file as the input file.

Note: To use the customized table, you must specify the name of the new table as the keyboard definition table in your HCON session profiles.

Related Information

The **e789kdef** command and **e789_ktbl.p** file format.

The **/usr/lib/hcon/e789_ktbl** file.

How to Customize the HCON Color Definition Table on page 4–59.

How To Define a New Terminal for HCON

Prerequisite Tasks or Conditions

- HCON must be installed on your system.
- You must have read access to the **/usr/lib/terminfo** directory.
- You must have read and write access to the **/usr/lib/hcon/terminfo** directory.
- You should be familiar with using the **terminfo** facility to create terminal definitions.

Procedure

When installed, HCON supports all RISC System/6000 high function terminals (HFTs), and the following ASCII terminals: IBM 3151, IBM 3161, IBM 3162, IBM 3163, DEC VT100, DEC VT220, and WYSE WY–50.

The **terminfo** source files (the **ibm.ti.H**, **dec.ti.H**, and **wyse.ti.H** files) used to create these supported terminals are in the **/usr/lib/hcon/terminfo** directory. You can look at these files to see an example of a terminal definition. The **terminfo** binary files for these supported terminals are in subdirectories of the **/usr/lib/hcon/terminfo** directory. Each subdirectory is named with the first letter of the terminal file name. For example, the IBM 5081 **terminfo** binary file, **ibm5081.H**, is in the **/usr/lib/hcon/terminfo/i** directory.

In order to run HCON on another type of terminal, it is necessary to use the AIX **terminfo** facility to map the new terminal keys to the HCON key names.

1. Create a **terminfo** source file in the **/usr/lib/hcon/terminfo** directory.
   a. Put a **terminfo** entry for your new terminal in the file. The name of the **terminfo** entry must be the name of your terminal with a **.H** appended to it. For example, if the terminal you are adding is an XYZ2500, use `XYZ2500.H` for the entry name.
   b. If your **terminfo** entry uses definitions from other **terminfo** source files (by including them with the **terminfo use** capability), you must include the contents of those files in your source file, preceding the entry for the new terminal.
2. Set the **TERMINFO** environment variable to **/usr/lib/hcon/terminfo**.
3. Execute the **tic** command to create your new terminal definition.
4. To use your new terminal definition, set the **TERM** environment variable to the new terminal name (without the **.H**).
5. If necessary, create a new keyboard definition table for HCON and map keys on your new keyboard to the emulator functions.

Related Information

The List of HCON and terminfo Key Names on page 4–62 shows how **terminfo** key names correspond with HCON key names.

The **termdef** command, **tic** command.

How to Customize the HCON Keyboard Definition Table on page 4–60 discusses the steps involved in mapping keyboard keys to HCON functions.

List of HCON and terminfo Key Names

The following table shows how the **terminfo** capability names are mapped to HCON key names:

HCON Key Name	terminfo Name	HCON Key Name	terminfo Name
key_tab	ktab	key_pf6	kf6
key_btab	kbtab	key_pf7	kf7
key_enter	kcmd	key_pf8	kf8
key_sysreq	khlp	key_pf9	kf9
key_clear	kclr	key_pf10	kf10
key_backspace	kbs	key_pf11	kf11
key_chgdis	ksel	key_pf12	kf12
key_insert	kich1	key_pf13	kf13
key_delete	kdch1	key_pf14	kf14
key_home	khome	key_pf15	kf15
key_pa1	kpp	key_pf16	kf16
key_pa2	knp	key_pf17	kf17
key_pa3	kll	key_pf18	kf18
key_spaceof	kend	key_pf19	kf19
key_down	kcud1	key_pf20	kf20
key_ddown	kc3	key_pf21	kf21
key_right	kcuf1	key_pf22	kf22
key_rright	ka3	key_pf23	kf23
key_left	kcub1	key_pf24	kf24
key_lleft	kcl	key_print	kf0
key_up	kcuu1	key_saves	kil1
key_uup	ka1	key_repls	kdl1
key_pf1	kf1	yen_sign	kind
key_pf2	kf2	overbar	kel
key_pf3	kf3	english_pound	kri
key_pf4	kf4	cent_sign	ked
key_pf5	kf5		

Related Information

How to Customize the HCON Keyboard Definition Table on page 4–60, How to Define a New Terminal for HCON on page 4–61.

How to Create HCON File Transfer Shell Procedures

Prerequisite Condition

- To run the file transfer shell procedure, you must be registered as an HCON user.

Procedure

An HCON file transfer can be invoked in a shell procedure simply by entering the user interface commands into a file and running the procedure. The procedure can inspect the return code from the **fxfer** command by testing the **$?** environment variable after the command runs. A return code of 1 indicates that an error has occurred, and a return code of 0 (zero) indicates that the transfer has successfully completed.

Sample Shell Procedure

The following shell procedure accepts the name of the RISC System/6000 file as the first command line parameter (the **$1** procedure variable) and the name of the host file as the second parameter (the **$2** procedure variable). It instructs the **fxfer** command to:

- Perform an implicit logon if necessary (if session `a` is not already established on the RISC System/6000) and log into the host as user `matthew`, using the `SYSvm6` AUTOLOG script.
- Upload the RISC System/6000 file to the host.
- Perform ASCII-to-EBCDIC translation. The file transfer process translates the file using the table specified by the country parameter in the session profile.
- Place messages from the file transfer command in the `$HOME/transfer.stat` file.

The shell procedure checks the error code environment variable (the **$?** variable) to see if the transfer is successful and echoes this information to the user.

Notes:

1. The user running the shell procedure must include the appropriate quotation marks when entering the host file name.
2. The file transfer process will prompt the user for the host password.

Following is the `fx` sample shell procedure:

```
## fx shell procedure  ##
# This procedure accepts two command line parameters:
#   $1 – local file name
#   $2 – host file name
#
# This procedure uploads the local file to the host specified
# in HCON session profile a, and translates the file from
# ASCII to EBCDIC.
#
# If necessary, the procedure logs into the host implicitly.
# Status messages are placed in the $HOME/transfer.stat file.
# After the procedure runs, the file transfer prompts for
# the host password.
#
fxfer –n a –utf $HOME/transfer.stat –x matthew,vm6 $1 $2
if test $? != 0
then
  echo "File transfer return value indicates an error."
  echo "See the $HOME/transfer.stat file for information."
else
  echo "File transfer completed successfully."
fi
```

Using the File Transfer Procedure

To use the `fx` shell procedure to transfer the local file `/u/matthew/newinfo` to the host file `newinfo1` on an MVS/TSO host, enter:

```
fx /u/matthew/newinfo "newinfo1"
```

The procedure invokes the **fxfer** command to transfer the file to the host computer and prompts for the host password.

To use the shell procedure for a transfer to a VM/CMS host, enter:

```
fx /u/matthew/newinfo "newinfo1 file a"
```

In each case, the host type must be specified in the session profile.

Avoiding the Password Prompt

To avoid being prompted for the password, redirect input into the file transfer. For example, if user `matthew`'s host password is `u1matt`, change the file transfer code in the procedure as follows:

```
fxfer –n a –utf $HOME/transfer.stat –x matthew,vm6 $1 $2 << END
u1matt
END
```

Using this redirection allows the procedure to bypass prompting the user for the login ID and password. Instead, the file transfer process uses the password `u1matt`.

Similarly, you can use the **echo** command to give the password to the shell procedure, as follows:

```
echo "u1matt" | fx /u/matthew/infofile "infofile1"
```

Another way to avoid being prompted for the password is to place the host password in a file and use the **cat** command to give the password to the shell procedure. For example, if the password is in the file `/u/matthew/logfile`, enter the following to start the file transfer:

```
cat /u/matthew/logfile | fx /u/matthew/newinfo "newinfo1"
```

Note: When using these techniques to avoid a password prompt, you may also want to specify a status file using the **–f** flag of the **fxfer** command. Otherwise, all of the file transfer status messages are put in the **$HOME/hconerrors** file.

Using the Same Shell Procedure with Different Host Login IDs

Using a separate file for the host password also allows you to change the login ID for the host. For example, change the file transfer code in the procedure as follows:

```
fxfer –n a –utf $HOME/transfer.stat $1 $2
```

Then include both the host login ID and the host password in a separate file and use the **cat** command to give the information to the file transfer procedure. For example, if the `/u/matthew/.hostinfo` file contains the following lines:

```
matt,vm6
mattpass
```

the command

```
cat /u/matthew/.hostinfo | fx /u/matthew/upfile "newfile"
```

causes the `fx` shell procedure to invoke the **fxfer** command and log into the specified host using the login ID string `matt,vm6` and the host password `mattpass`.

Related Information

The **fxfer** command transfers files between a RISC System/6000 and a System/370 host.

Understanding the HCON File Transfer Programming Interface in *Communications Programming Concepts* discusses writing programs with the HCON file transfer programming interface, which consists of the **fxfer** function and **cfxfer** function.

AUTOLOG and LAF routines are used to perform implicit logons to the host. Understanding HCON Explicit and Implicit Logon in *Communications Programming Concepts* discusses these routines.

To start an emulator session explicitly, use the **e789** command.

Shells Scripts in the Shells Overview in *General Programming Concepts* discusses developing shell procedures.

Understanding HCON File Transfers on page 4–31 describes the HCON file transfer process.

Part 2. Network Facilities

Chapter 5. Network Overview

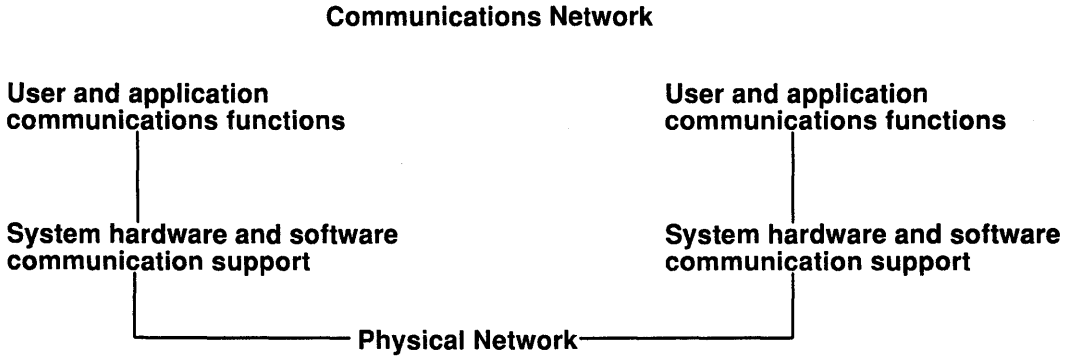
The Network Overview chapter provides background information necessary for an understanding of network–related topics. It contains brief explanations of concepts specific to communications networks. Types of networks are discussed as well as features common to the different types. This chapter briefly discusses the various kinds of applications, or programs, that are available for networks.

Networks and Communications

Network is a widely used, often ambiguous term that commonly takes its meaning from the context in which it is used. This network overview distinguishes three general levels:

- User and application communications functions at the top level
- System hardware/software communications support at the middle level
- Physical networks at the bottom level.

Combined, these levels form a communications network.



People and organizations use networks for many reasons. The following reasons are a few possibilities:

- Data entry
- Inquiry
- Record updating
- Remote batch entry
- Share resources
- Share data
- Electronic mail.

Data entry consists of entering data directly into either local or remote data files, reducing the need for such intermediate steps as posting, recording, or punching. Increased accuracy and efficiency are natural by-products of a one-step data transfer. Inquiry allows for searching data files for specified information. Record updating consists of altering, adding, or deleting data stored in local or remote files. Remote batch entry consists of entering

batches of data from a remote location. Because of such diverse capabilities, communications and networks are not only desirable but necessary.

Sharing resources is another function of physical networks. Not only can users share data, they can also share programs, file-storage space, and peripheral devices (for example, printers, modems, terminals, hard disks, teletype machines). In a distributed environment, it is often prohibitive to have a printer for each user or, because of limited memory space, to have the same files residing at each node.

The communications network creates an economical and efficient situation whereby many users can share files stored on one file server or share the use of one printer or modem for data transfer. Electronic mail allows two users on different systems to communicate across a physical network.

Physical Networks

The physical network consists of the cable (Standard Ethernet, IEEE 802.3 Ethernet, Token-Ring, telephone lines) that connects the different hardware residing on the network.

Physical networks vary both in size and in the type of hardware used.The three common kinds of networks are *local area networks* (LANs), *wide area networks* (WANs), and *asynchronous networks* (ASYNCs). A LAN is a network in which communications are limited to a moderately sized geographic area (1 to10 km), such as a single office building, warehouse, or campus.

A WAN is a network providing data communication capability in geographic areas larger than those serviced by LANs; for example, across a city, state, or country. An ASYNC network is characterized by data transmissions in which transmission of a character or a block of characters can begin at any time but in which the bits that represent the character or block have equal time duration.

LANs commonly use Standard Ethernet, IEEE 802.3 Ethernet, or Token-Ring hardware for the physical network, while wide area and asynchronous networks commonly use communications networks that are provided by communications common carrier companies. Operation of the physical network is usually controlled by networking standards (for example, EIA232D or CCITT V.35). In general, for most users, it is not necessary to understand how the networking equipment works to use a data communications system.

System Hardware and Software Communication Support

Common to all communications that use networks is the use of hardware and software. *Hardware* consists of the physical equipment connected to the physical network. *Software* consists of the programs, procedures, rules or protocols, and associated documentation pertaining to the operation of a particular system. The hardware and software communications support is defined both by the hardware being used and the software necessary to run that hardware and interface with the network.

The system hardware consists of adapter cards that provide a path or interface between the system software and the physical network. An adapter card requires an input/output (I/O) card slot in the system. It is this card that connects the *data terminal equipment* (DTE) to the *data circuit terminating equipment* (DCE); that is, it provides physical local addressing to a DTE port.

An adapter card prepares all inbound and outbound data, performs address searches, provides drivers, receivers, and surge protection, supports different interfaces, and in general relieves the system processor of many communications tasks. Adapter cards support the standards required by the physical network (for example, EIA232D, Smart

Modem, V.25 bis, EIA422A, X.21, or V.35) and at the same time support software *protocols* (for example, SDLC, HDLC, and Bisynchronous protocols)

Protocols

All communications software uses protocols, sets of semantic and syntactic rules that determine the behavior of functional units in achieving communication. Protocols define how information is delivered, how it is enclosed to reach its destination safely, and what path it is going to follow. Protocols also coordinate the flow of messages and their acknowledgments; for example, whether a response is expected.

There are also conversation protocols regarding the priority of talking over a system so that information isn't lost. It is important to know who is talking at the moment and who will talk next. When users talk at the same time, information can be lost. Finally, there are protocols regarding how data is formatted and presented. Do the sender and receiver use the same encoding? Do they speak the same language, or is translation needed?

Protocols exist at different levels within the kernel and cannot be manipulated directly; however, they are manipulated indirectly by what the user chooses to do at the Application Programming Interface (API) level. The choices a user makes in invoking file transfer, remote login, or terminal emulation programs will define the protocols used in the execution of those programs.

Addresses

A third feature common to communications networks is *addresses*. Addresses are associated with both software and hardware. The address is the means by which the sending or control station selects the station to which it sends data. In effect, addresses are a means of identifying receiving or storage locations. A physical address is a unique code assigned to each device or workstation connected to a network.

For example, on a Token-Ring network, the **netstat –iv** command displays the token–ring card address. This is the physical network address. The **netstat –iv** command also displays class-level and user-level address information. Addresses are often defined by software but can be created by the user as well.

Domains

An aspect of addresses common to many communications networks is the concept of *domains*. For example, the structure of the Internet illustrates how domains define the Internet Protocol (IP) address. The Internet is a large network that comprises many different smaller networks. To facilitate routing and addressing, Internet addresses are hierarchically structured in domains, with very broad categories at the top such as `com` for commercial users, `edu` for educational users, and `gov` for government users.

Within the `com` domain are many smaller domains corresponding to individual businesses; for example, `ibm`. Within the `ibm.com` domain are even smaller domains corresponding to the Internet addresses for the various locations of IBM, such as `austin.ibm.com` or `raleigh.ibm.com`. At this level, we start seeing names of *hosts*. A host, in this context, is any computer connected to the network. Within `austin.ibm.com` there may be hosts with the names `hamlet` and `lear`, which are addressed `hamlet.austin.ibm.com` and `lear.austin.ibm.com`.

Gateways and Bridges

A wide variety of networks reside on the Internet, often using different hardware and running different software. *Gateways* and *bridges* enable these different networks to communicate with each other. Bridges are used on *local area networks* (LANs). A bridge is a functional unit that connects two LANs that use the same logical link control (LLC) procedure but possibly different medium access control (MAC) procedures. A gateway has a broader range than a bridge. It operates above the link layer and translates, when required, the interface and protocol used by one network into those used by another distinct network. Gateways allow data transfers across the various networks that comprise the Internet.

Routing

Using domain names for addressing and gateways for translation greatly facilitates the *routing* of the data being transferred. Routing is the assignment of a path by which a message reaches its destination. The domain name effectively defines the message destination. In a large network like the Internet, information is routed from one communications network to the next until that information reaches its destination. Each communications network checks the domain name and, based on the domains with which that network is familiar, then routes the information on to the next logical stop. In this way, each communications network that receives the data contributes to the routing process.

Local and Remote Nodes

A physical network is used by the hosts that reside on that network. Each host can be said to be a *node* on the network. A node is an addressable location in a communications network that provides host processing services. The inter-communication of these various nodes gives rise to the concepts of *local* and *remote*. Local pertains to a device, file, or system accessed directly from your system, without the use of a communications line. Remote pertains to a device, file, or system accessed by your system over a communications line. Local files reside on your system, while remote files reside on a file server or at another node with which you communicate using a physical network (for example, Ethernet, Token Ring, phone lines).

Server and Client

Related to the concepts of local and remote are those of *client* and *server*. A server is a computer that contains data or provides facilities to be accessed by other computers on the network. File servers, which store files, name servers, which store names and addresses, and code servers, which store programs and applications, are common server types.

A client is a computer requesting services or data from a server. A client, for example, could request updated program code or the use of applications from a code server. To obtain a name or address, a client contacts a name server. A client could also request files and data for data entry, inquiry, or record updating from a file server.

User and Application Communications Functions

Networks allow for several user and application communications functions. They enable a user to:

- Send electronic mail
- Emulate another terminal or login to another computer.
- Transfer data
- Execute programs residing on a remote node.

The communications network also enables one user to communicate with another user, in this case using a form of electronic mail. The underlying layers of software and hardware as well as the physical network allow a user to generate, send, receive, and process messages, letters, memos, invitations, and data files to or from any other user who resides on the physical network. Electronic mail has the capability for message annotation, message sequencing, message packing, date sorting, and mail folder capabilities.

Emulating another computer permits users to access applications in other systems as if directly attached to that system. Remote login allows remote login between an AIX-based system and other UNIX-based systems.

A file transfer protocol allows users to access files and directories on remote hosts. Files may be transferred to and from the remote host. Password protection is usually provided as part of the protocol. With a file transfer, there is a client/server relationship between the user initiating the request and the remote system the user is accessing. Often a file transfer protocol includes functions for display and control so users with read/write access can display, define, or delete files and directories.

Several different protocols have been devised to allow users and applications on one system to invoke procedures and application execution on other systems. This can be useful for a number of environments, including the offloading of many computer-intensive routines in engineering and scientific applications.

DOS Server Program Requirements

Different types of computers can be connected on a network. That is, the computers can be from different manufacturers or be different models from the same manufacturer. The differences in operating systems of two or more types of computers can be bridged with programs.

Sometimes these programs require that another program already be installed on the network. For example, the DOS Server Program requires that the IBM AIX for DOS Users program be running on the IBM Personal Computers or PS/2s connected to a network. Other programs may require that such communications connectivity protocols as TCP/IP or SNA exist on the network.

Sometimes to allow the programs to run, a file must be uncommented in a start up file. To uncomment a line in a start up file, the # (pound) sign must be removed from the beginning of the line. For example, if you have purchased IBM AIX Access for DOS Users, uncomment the line in the **/etc/rc** file that calls the **/etc/rc/pci** shell script. The DOS Server program may then be started by either executing the shell script or by restarting your system.

Related Information

TCP/IP Overview for System Management on page 14–1, BNU Overview for System Management on page6–1 , X.25 Overview for System Management on page 15–1, NFS Overview for System Management on page 11–1, Mail Overview for System Management on page 8–28.

General Information and Planning Information Manual

Chapter 6. Basic Networking Utilities (BNU)

The Basic Networking Utilities (BNU) are a version of the UNIX-to-UNIX Copy Program. They can be used to communicate with any AIX or UNIX system on which BNU or another version of UUCP is running. This chapter provides conceptual and procedural information to help you set up and run BNU on your system, including examples of BNU configurations for three different types of network connections. Also in this chapter are lists of the BNU commands, directories, files, and file formats.

BNU Introduction for System Management

The Basic Networking Utilities (BNU) are a group of programs, directories, and files that can be used to communicate with any UNIX system on which a version of the UNIX-to-UNIX Copy Program (UUCP) is running. Information about managing BNU is divided into three major sections:

- Read the BNU Overview for System Management on page 6–1 to find out about the parts of BNU and how they work on your system.
- Read the BNU Overview of System Management Tasks on page 6–12 to find out about the tasks that need to be completed so that BNU can be used on your system.
- Read the List of Information for BNU System Management on page 6–18 for a list of reference information about managing BNU.

For general information about networks, read the Network Overview on page 5–1.

BNU Overview for System Management

The set of AIX Basic Networking Utilities (BNU) programs is a version of the standard UNIX-to-UNIX Copy Program (UUCP). AIX BNU can be used to communicate with any UNIX system on which a version of UUCP is running.

The Basic Networking Utilities that are included with the Extended Services for the AIX Base Operating System are based on AT&T System V BNU, also known as HoneyDanBer UUCP. However, AIX BNU contains some Berkeley System Distribution (BSD) commands as well.

BNU uses a set of hardware connections and software programs to communicate between systems. Using BNU, you can communicate with systems attached to your own system in several different ways: with direct connections, modem connections, or connections over a TCP/IP interface. In addition, you can use a system connected directly to your system as a relay station to reach other UNIX systems not connected to your system. In this way, UUCP provides a network of systems all over the world.

BNU can perform file transfers and send mail between your system and other systems running versions of UUCP. In addition, BNU can execute commands on remote systems and log in to remote systems.

BNU uses a structure of directories and files to keep track of its activities. This structure includes a set of public directories, a group of administrative directories and files, configuration files, and lock files. Most of the directories for BNU are created during the installation process. Some of the administrative directories and files are created by various BNU programs as they run. The way BNU works on your system is controlled by the configuration files, which interact with each other to determine whether remote systems can log in to your system and what they can do after they log in.

With the exception of the remote login commands, BNU works as a batch system. When a user requests a job sent to a remote system, BNU stores the information it needs to complete the job. This is known as *queuing* the job. At certain prescheduled times, or when a user instructs it to do so, BNU contacts various remote systems, transfers the work queued up for them, and accepts jobs from them. The times of these transfers and how the transfers work are controlled by the configuration files on your system and those of the remote system.

Because other systems can also contact your system to log in, transfer files, and execute commands, BNU provides a means to establish security. BNU security allows you to restrict what users of remote systems can do on your system, and allows users of remote systems to restrict what you can do on their systems.

BNU runs several daemons to complete its activities. Once BNU is in use, it uses the set of administrative directories and files to store the files it needs and log its own activities.

For Japanese language support:

> All BNU commands except the **uucpadm** command are available for Japanese language support. User names need not be in ASCII characters. However, all system names and file names must be in ASCII characters. If a user attempts to schedule a transfer or remote command execution involving non-ASCII system names or file names, BNU returns an error message.

The remainder of this overview is divided into three major sections, as follows:

- Understanding BNU Security on page 6–3 discusses BNU security.
- Understanding the BNU Daemons on page 6–5 discusses the BNU daemons.
- Understanding the BNU File and Directory Structure on page 6–7 discusses the files BNU uses for configuration and administration.

Read the BNU Overview of System Management Tasks on page 6–12 to find out about the tasks that need to be completed so that BNU can be used on your system.

The List of Information for BNU System Management on page 6–18 lists the conceptual, procedural, and reference information about managing BNU.

Related Information

List of BNU User Commands on page 6–19 lists the BNU commands available to all AIX users.

The tip Overview for System Management on page 6–10 discusses the **tip** command, which is similar to the BNU **cu** command.

Understanding BNU Security

Because other systems can contact your system to log in, transfer files, and execute commands, BNU includes a means to establish security. BNU security allows you to restrict what users of remote systems can do on your system and allows users of remote systems to restrict what you can do on their systems.

BNU security works on several levels. When you configure BNU, you can determine the following:

- Who on your system has access to BNU files.
- Which remote systems users of your system can contact.
- How users on remote systems log in to your system.
- What users on remote systems can do on your system once they log in.

The uucp Login ID

When BNU is installed, all of the configuration files, the BNU daemons, and many of the BNU commands and shell procedures are owned by the **uucp** login ID. The **uucp** login ID is a special login ID, with a UID of 5. It is a member of the uucp group, which has a GID of 5. The **cron** daemon reads the **/usr/spool/cron/crontabs/uucp** file to schedule automatic jobs for BNU.

Usually, logging in as user **uucp** is not allowed. To change files that are owned by the **uucp** login ID, you must log in as a user with root user authority.

Warning: Allowing remote systems to log in to the local system with the **uucp** login ID seriously jeopardizes the security of your system. Remote systems logged in with the **uucp** ID can display and possibly modify (depending on the other permissions specified in the LOGNAME entry) the local **Systems** and **Permissions** files. It is strongly recommended that you create other BNU login IDs for remote systems and reserve the **uucp** login ID for the person responsible for administering BNU on the local system. For the best security, each remote system that contacts the local system should have a unique login ID with a unique UID number.

BNU Login IDs

Login IDs used by remote systems should have passwords. The startup shell for these IDs should be the **uucico** daemon (**/usr/lib/uucp/uucico**). Thus when remote systems call your system, they automatically start the **uucico** daemon on your system. Login IDs for BNU should belong to the uucp group, with a GID of 5.

In order to prevent AIX security from prompting a new BNU login ID for a new password when the remote system logs in, you must set the password as soon as you create the account. To do this, use the **pwdadm** command. For example, to set a password for the login ID `nuucp`, log in as the root user and enter the following command:

```
pwdadm -f NOCHECK nuucp
```

The system prompts you for the root password for verification, then for a password for the `nuucp` login ID. Completing these steps allows the remote system to log in without being immediately prompted for a new password.

After creating the login ID for a remote system, notify that system's BNU administrator of the login ID and password necessary to access your system.

Creating a BNU Administrative Login ID

Someone with root user authority can also set up a BNU administrative login ID. This is useful if, for example, security requirements on your system prevent the BNU administrator from having root user authority. The BNU administrative login ID should have password security, a UID of 5, and be in the uucp group (with a GID of 5). The login shell for the administrative login should be the **/bin/sh** program (instead of the **uucico** daemon). Giving the BNU administrative login a UID of 5 causes it to have the same privileges as the **uucp** login ID. Thus, for security, remote systems should not be allowed to login as the BNU administrator.

Security and the Systems and remote.unknown Files

On most BNU systems, only remote systems listed in the **/usr/lib/uucp/Systems** file can log in to your system. The **/usr/lib/uucp/remote.unknown** script is executed whenever an unknown system attempts to call the local system. As delivered, this script refuses to let the unknown system log in, and makes an entry in the **/usr/spool/uucp/.Admin/Foreign** file recording the time of the login attempt.

Someone with root user authority, or a BNU administrator, can modify the **remote.unknown** shell procedure if you wish to log more information about the remote system or to store the information in a different file. For example, you can modify the shell procedure so that it sends mail to the BNU administrator whenever an unknown system tries to log in.

By taking away execute permissions on the **remote.unknown** shell procedure, you can allow unknown machines to log in. In this case, you should add a MACHINE=OTHER entry to the **/usr/lib/uucp/Permissions** file to establish permissions for the unknown machines.

In any case, your system can contact only remote systems listed in the **Systems** file. This prevents users on your system from contacting unknown and untrusted systems.

Security and the Permissions File

The **/usr/lib/uucp/Permissions** file determines two things: the login IDs remote systems use when logging in to the local system and the commands and other privileges remote systems have on the local system.

The file contains two types of entries: LOGNAME and MACHINE entries. The LOGNAME entry defines login names and privileges associated with them. The MACHINE entry defines machine names and privileges associated with them. LOGNAME entries take effect when a remote system calls the local system and attempts to log in. MACHINE entries take effect when the local system contacts a remote system.

The options provided in the **Permissions** file allow you to establish various levels of security for each remote system. For example, it is possible (although not recommended) for many remote systems to share one login ID on the local system. However, you can also use the VALIDATE option to require each remote system to use a unique login ID. The SENDFILES, REQUEST, and CALLBACK options allow you to specify which system has control, keeping the local system in control of transactions if necessary.

The READ, WRITE, NOREAD, and NOWRITE options allow you to define access or lack of access to specific directories on the local system. These options allow you to control where on your system remote users can place data. Similarly, with the COMMANDS option you can limit the number of commands users on remote systems can execute on the local system.

However, you can also use the COMMANDS=ALL option to allow total privileges to systems closely associated with your system.

Warning: The COMMANDS=ALL option can seriously jeopardize the security of your system. Use it with extreme care.

For further information on the BNU configuration files and how they interact, read Understanding the BNU Configuration Files on page 6–8 and Configuring BNU on page 6–13.

Related Information

The **Permissions** file contains access permission codes, the **Systems** file lists accessible remote systems, and the **remote.unknown** shell procedure records contacts from unknown systems in the **/usr/spool/uucp/.Admin/Foreign** file.

The **pwdadm** command sets a password for a new user ID.

Understanding the BNU Daemons

The BNU software includes four daemons: the **uucico** daemon, **uuxqt** daemon, **uusched** daemon, and **uucpd** daemon. The daemons handle file transfers, remote command executions, work request scheduling, and communications using TCP/IP, respectively. The four BNU daemons are stored in the **/usr/lib/uucp** directory.

The **uucico**, **uusched**, and **uuxqt** daemons are normally started by the **cron** daemon according to a schedule set by the person who administers BNU. However, someone with root user authority can start these daemons manually, if necessary. The **uucpd** daemon should be started by the TCP/IP **inetd** daemon.

The uucico Daemon

The **uucico** daemon is the primary BNU daemon. It transports the files required to send data from one UNIX system to another UNIX system. The **uucp** and **uux** commands each start the **uucico** daemon to transfer command, data, and execute files to the designated system. The **uucico** daemon is also started periodically by the BNU scheduler, the **uusched** daemon, which handles the transfer of files queued in the local spooling directory.

Once started by the **uusched** daemon, the **uucico** daemon attempts to contact other systems and execute the instructions in the command files that have been randomized by the **uusched** daemon. To execute the instructions in the command files, the **uucico** daemon first checks the **/usr/lib/uucp/Systems** file to find an entry for the system to be called. Then it checks the **Systems** file entry to see if this is a valid time to call. If so, it checks the *Type* and *Class* fields and goes to the **/usr/lib/uucp/Devices** file to find a device that matches.

After finding a device, the **uucico** daemon checks the **/etc/locks** directory for a lock file for the device. If one exists, the daemon checks for another device of the requested type and speed, and uses it if available.

When no device is available, the daemon returns to the **Systems** file to find another entry for the remote system. If one exists, the daemon repeats the process of searching for a device. If none is found, the daemon makes an entry in the **/usr/spool/uucp/.Status/***SystemName* file for that remote system and goes on to the next request. The command file remains in the queue, and the **uucico** daemon attempts the transfer again later. The later attempt is called a *retry*.

When the **uucico** daemon reaches the remote system, it uses the instructions in the **Systems** file to log in. Logging in causes an instance of the **uucico** daemon to be invoked on the remote system as well.

The two **uucico** daemons, one on each system, work together to make the transfer. The **uucico** daemon on the calling system controls the link, specifying the requests to be performed. The **uucico** daemon on the remote system checks the local permissions to see whether they allow the request to be performed. If so, the file transfer starts.

After the **uucico** daemon on the calling system has finished transferring all the requests it has for the remote system, it sends a hangup request. When the remote **uucico** daemon has transactions to send to the calling system, it denies the hangup request, and the two daemons reverse roles.

Note: Either the **/usr/lib/uucp/Permissions** file on the local system or the **/usr/lib/uucp/Permissions** file on the remote system can forbid the daemons to reverse roles. In this case, the remote system must wait to transfer files until it calls the local system.

When nothing is left to be transferred in either direction, the two **uucico** daemons hang up. At this point, the **uuxqt** daemon is called to execute remote command requests.

Throughout the transfer process, the **uucico** daemons on both systems write messages in the BNU log and error files.

The uusched Daemon

The **uusched** daemon schedules the transfer of files that are queued in the spooling directory (the **/usr/spool/uucppublic** directory) on the local system. When the **uusched** daemon is invoked, it scans the spooling directory for command files, then randomizes the files and starts the **uucico** daemon, which actually transfers the files.

The uuxqt Daemon

The **uuxqt** daemon executes a command on a designated system. A user issues the **uux** command to run a specified command on a designated system. After creating the necessary files, the **uux** command starts the **uucico** daemon, which transfers those files to the public spooling directory on the specified system.

The **uuxqt** daemon on every connected system periodically searches the spool directory for command-execution requests. When it locates such a request, the **uuxqt** daemon checks for necessary files and permissions and then, if permitted, executes the specified command.

The uucpd Daemon

The **uucpd** daemon must be running on both the local and remote systems before BNU can establish communications with a remote computer by means of the Transmission Control Protocol/Internet Protocol (TCP/IP). The daemon is a subserver of the TCP/IP **inetd** daemon, and is started by the **inetd** daemon.

The **uucpd** daemon should be started at system startup if you use TCP/IP for BNU connections. However, the daemon can be started anytime before the connection between the two systems is established.

By default, the **inetd** daemon is configured to start the **uucpd** daemon. However, if this has been changed on your system, you may need to reconfigure the **inetd** daemon to start the **uucpd** daemon.

For further information on scheduling daemons, read Understanding the BNU Administrative Directories and Files on page 6–9.

Related Information

The **uucico** daemon, **uucpd** daemon, **uusched** daemon, **uuxqt** daemon.

The **cron** daemon, **inetd** daemon.

How to Set Up Automatic Monitoring of BNU on page 6–27 discusses setting up BNU schedules.

How to Configure BNU for use with TCP/IP on page 6–26 discusses starting the **uucpd** daemon.

Understanding the BNU File and Directory Structure

The Basic Networking Utilities use a structure of directories and files to keep track of their activities. This structure includes a set of public directories, a group of administrative directories and files, configuration files, and lock files. Most of the directories for BNU are created during the installation process. Some of the administrative directories and files are created by various BNU programs as they run.

The BNU public directory, **/usr/spool/uucppublic**, is the storage spot for files that have been transferred to the local system from other systems. The files wait in the public directory until users claim them with the **uupick** command or until the commands requested by users on remote systems have been executed by the **uuxqt** daemon. The public directory is created when BNU is installed. Within the public directory, BNU creates a subdirectory for each remote system that sends files to the local system.

The BNU administrative directories and files are in subdirectories of the **/usr/spool/uucp** directory. These directories and files contain two types of information: data waiting to be transferred to other systems, and log and error information about BNU activities.

The BNU configuration files reside in the **/usr/lib/uucp** directory. The files must be configured for your system. Three of them, the **Systems**, **Devices**, and **Permissions** files, are required before you can use BNU on the system. Setting up the other files is optional.

The configuration files contain information for BNU about remote systems you communicate with, devices you use to contact these systems, times to contact the systems, and what the remote systems are allowed to do on your system. Some configuration files also specify limits on BNU activities, to prevent your system from becoming overloaded.

The lock files that BNU uses are stored in the **/etc/locks** directory. Whenever BNU uses a device to connect to a remote computer, it places a lock file for that device in the **/etc/locks** directory. When another BNU program or any other program needs the device, that program checks the **/etc/locks** directory for a lock file. If a lock file exists, the program waits until the device is available or uses another device for the communication.

In addition to device lock files, the **uucico** daemon places lock files for remote systems in the **/etc/locks** directory. Before contacting a remote system, the **uucico** daemon checks the **/etc/locks** directory for a lock file for that system. These files prevent other instances of the **uucico** daemon from establishing duplicate connections to the same remote system.

Note: Other AIX systems besides BNU, such as ATE and TCP/IP, use the **/etc/locks** directory.

Related Information

List of BNU Files and Directories on page 6–20.

The **/usr/spool/uucppublic/*** directories.

The **/etc/locks** directory.

Understanding the BNU Configuration Files

The BNU Configuration files, also known as the BNU Supporting Database, are stored in the **/usr/lib/uucp** directory. They are owned by the **uucp** login ID and can be edited only by someone with root user authority. These files determine which systems your system can talk to, when and how often your system contacts other systems, which systems can contact your system and what they can do on your system, and the hardware connections and parameters your system uses to contact other systems.

The entries you put in the BNU Configuration files depend on the types of connections between your system and each remote system. Special entries must be made in several of the files if you are using TCP/IP or direct connections to contact other systems. If you are using modems to contact other systems, the modems must be defined in the **Dialers** file.

The **Systems**, **Devices**, and **Permissions** files must be set up on your system before you can contact remote systems using BNU. The other files must be set up for you to fully use some BNU capabilities, such as automatic polling. Also, the various configuration files must be modified periodically to reflect changes to your system or the systems you contact.

The BNU configuration files are:

Devices	Contains information about available devices, including both modems and direct connections.
Dialcodes	Contains dialing code abbreviations, which allow you to shorten phone numbers in the **Systems** file.
Dialers	Specifies initial handshaking on a connection.
Maxuuscheds	Limits simultaneous scheduled jobs.
Maxuuxqts	Limits simultaneous remote command executions.
Permissions	Contains access permission codes. This file is the primary file for determining the security for BNU.
Poll	Specifies when the BNU program should poll remote systems to initiate tasks.
remote.unknown	Records contacts from unknown systems (those not listed in the **Systems** file).
Systems	Lists accessible remote systems and information needed to contact them, including the device to use and the password you need to log in. Also specifies the times when the systems can be contacted.

The configuration files interact when BNU is in use. For example:

- The **Devices** file contains a *Token* field that refers to entries in the **Dialers** file.
- The **Systems** file contains an entry for a *Class* of device. A device of each *Class* referred to in the **Systems** file must be defined in the **Devices** file.
- The **Poll** file contains entries for systems your system calls. Each of these systems must be defined in the **Systems** file.

Related Information

Example of a BNU Configuration for a Telephone Connection on page 6–37, Example of a BNU Configuration for a TCP/IP Connection on page 6–34, and Example of a BNU Configuration for a Direct Connection on page 6–40 provide sample configurations for BNU.

Understanding the BNU Administrative Directories and Files

The Basic Networking Utilities use a set of administrative directories and files to store data and log BNU activities. The BNU administrative system is contained in the **/usr/spool/uucp** directory. Under this directory, BNU creates the following directories:

- The **.Admin** directory contains four administrative files: the **audit**, **Foreign**, **errors**, and **xferstats** files. These files contain error and log information about BNU activities.
- The **.Corrupt** directory contains copies of files that cannot be processed by the BNU program.
- The **.Log** and **.Old** directories contain log files from BNU transactions.
- The **.Status** directory stores the last time the **uucico** daemon tried to contact remote systems.
- The **.Workspace** directory holds temporary files that the file transport programs use internally.
- The **.Xqtdir** directory contains execute files with lists of commands that remote systems can run.
- Each *SystemName* directory contains files used by file transport programs. These are the Command (**C.***), Data (**D.***), Execute (**X.***), and Temporary (**TM.***) files. BNU creates a *SystemName* directory for each remote system it contacts.

The directories whose names begin with a dot are *hidden*. This means that they cannot be found with an **ls** or **li** command unless the **–a** flag is used. When the **uucico** daemon is started, it searches the **/usr/spool/uucp** directory for work files and transfers the files from any directory that is not hidden. Thus the **uucico** daemon sees only the *SystemName* directories, and not the other administrative directories.

The files in the hidden directories are owned by the **uucp** login ID. These files can generally be accessed only by someone with root user authority or by someone whose login ID has a UID of 5. The work files in the *SystemName* directories are owned by the users who scheduled the work.

For further information about maintaining the administrative directories, read Maintaining BNU.

Related Information

The **ls** command, **li** command.

The **uucico** daemon.

tip Overview for System Management

The **tip** command can be used to contact any system that is running the UNIX operating system and to which the local system is physically connected. The **tip** command is installed with the Basic Networking Utilities (BNU) and can use the same physical connections that are used by BNU, including TCP/IP connections, direct connections, and modem connections.

The **tip** command uses variables and escape signals, as well as flags, to control its operations. The flags can be entered at the command line. The escape signals can be used over a connection with a remote system to start and stop file transfers, change the direction of a file transfer, exit to an AIX subshell, and so on.

The **tip** command variables define settings such as the end-of-line character, the break signal, and the mode of file transfers. Variable settings can be initialized at run time using a **.tiprc** file. Variable settings can also be changed during execution using the **~s** escape signal. Some variables, such as the end-of-line character, can be set for an individual system in that system's entry in the **remote** file.

The **tip** command reads three files, the **phones** file, the **remote** file, and the **.tiprc** file, to determine initial settings for its variables. The **.tiprc** file must always be named the **$HOME/.tiprc** file. The names and locations of the **remote** and **phones** files can vary. The names of the **remote** file and the **phones** file can be determined by environment variables as follows:

PHONES Names the **phones** file used by the **tip** command.

REMOTE Names the **remote** file used by the **tip** command.

To use an environment variable, set it before starting the **tip** command. As an alternative, the names of the **phones** and **remote** files can be determined using the **tip** command **phones** and **remote** variables in the **.tiprc** file. If no variables are set to determine the names of these files, the command uses the **/etc/phones** file and the **/etc/remote** file, respectively.

Note: The **tip** command reads only the *last* **remote** or **phones** file specified. Thus, if you specify a **remote** or **phones** file with a variable, the new file is used in place of (not in addition to) any previous files you specified.

The **tip** command uses variable settings in the following order:

1. The command checks the settings of the **PHONES** and **REMOTE** environment variables to find the files to use for the **phones** and **remote** files, respectively.
2. The command reads the **.tiprc** file and sets all variables accordingly. If the **remote** or **phones** variables are set in the **.tiprc** file, these settings override the environment variable settings.
3. When a connection to a remote system is initiated, the command reads the **remote** file entry for that system. The settings in the **remote** file entry override settings made in the **.tiprc** file.
4. If the *–BaudRate* flag is used with the **tip** command, the specified rate overrides all previous baud rate settings.
5. A setting made with the **~s** escape signal overrides all previous settings of a variable.

Setting Up the tip Command

Before the **tip** command can connect to a remote system, the **/etc/remote** and **/etc/phones** files must be set up. The **/etc/remote** file defines attributes of remote systems such as the port and type of device to use to reach the system, as well as the signals to use to indicate the beginnings and endings of transmissions. The **/etc/phones** file lists telephone numbers used to contact remote systems over a modem line.

Sample **remote** and **phones** files are delivered with AIX BOS Extensions 1. The sample **remote** file is named **/etc/remote–file**. The sample **phones** file is named **/etc/phone–file**. To set up one of these files, copy the sample file to the correct name and modify it to suit the needs of your site.

Note: You must have root user authority to create files in the **/etc** directory.

Any **tip** user can create a **.tiprc** file and use this file to specify initial settings for **tip** variables. The **.tiprc** file must be placed in the user's **$HOME** directory.

A **tip** user can also create customized **remote** and **phones** files. An individual **remote** file must be in the format of the **/etc/remote** file and specified with the **tip** command **remote** variable or the **REMOTE** environment variable. An individual **phones** file must be in the format of the **/etc/phones** file and specified with the **tip** command **phones** variable or the **PHONES** environment variable. If an individual **phones** or **remote** file is specified with one of the variables, that file is read *in place of* (not in addition to) the **/etc/phones** or **/etc/remote** file.

Users of **tip** can use combinations of individual **phones** and **remote** files. For example, a user could read the default **remote** file, **/etc/remote**, but use an individual **phones** file named with the **phones** variable.

Related Information

The **tip** command is installed with BNU/UUCP, which is an AIX Optional Program Product. See How to Install AIX for RISC System/6000 Licensed Program Optional Program Products in *General Concepts and Procedures*.

BNU Overview for System Management on page 6–1 discusses BNU connections between systems.

List of Information about the tip Command

Read the following to find out more about the **tip** command:

tip command
tip Command Escape Signals
tip Command Variables.

The following file formats are used to configure the **tip** command:

phones file format
remote file format
.tiprc file format.

BNU Overview of System Management Tasks

Before users on your system can run BNU programs, BNU must be installed and configured.

The way BNU works on your system is controlled by a set of configuration files that interact with each other to determine whether remote systems can log in to your system and what they can do after they log in. These files must be set up according to the requirements and resources of your system.

BNU must be maintained. To maintain BNU, you must read and remove log files periodically and check the BNU queues to be sure jobs are transferring to remote systems properly. You must also periodically update the configuration files to reflect changes in your system or the systems it contacts.

The remainder of this overview is divided into three sections, as follows:

- Installing BNU on page 6–12
- Configuring BNU on page 6–13
- Maintaining BNU on page 6–14.

Related Information

List of Information for BNU System Management on page 6–18.

Installing BNU

The Basic Networking Utilities use both hardware and software to communicate with other systems. In order to use BNU on your system, BNU-related hardware must be installed, and the BNU software must be installed and configured.

Hardware for BNU

Before your system can communicate with another, a two-way communication link must be established between the systems. There are three ways to set up such a link:

- Using a hardwired line with a device such as `tty1`. This is known as a *direct* connection.
- Using a telephone line with a modem.
- Using a TCP/IP connection over a Token Ring, Ethernet, SL/IP, or X.25 network.

A direct connection links a port on the local computer to a port on the remote computer. A direct line is advantageous when users on local systems communicate frequently with remote systems. The link is always available and access time is short. However, a port used for a hardwired communications link is not available for any other purpose.

The second type of connection uses a telephone line and a modem. The user on the calling computer establishes the connection to a remote computer through an Automatic Calling Unit (ACU), also referred to as an autodialer or a modem. The modem attached to the called system answers the telephone, and the communications software then completes the connection.

The third type of connection uses TCP/IP. If your system is hooked into a TCP/IP network, you can set up BNU to allow your system to communicate with any other system in that network.

BNU Software

The Basic Networking Utilities software is composed of the following items:

- Directories in which the various files and programs for BNU are stored
- Configuration files containing information needed to establish remote connections and determine access permissions
- Administrative files used primarily in transferring data between computers
- User commands that perform the basic BNU functions, such as file transfer and remote command execution
- Administrative commands, such as programs that enable you to configure and maintain BNU
- Daemon programs that handle file transfers, communications with TCP/IP, scheduling of work, and remote command executions.

All of these items are installed during the software installation process. After both the software and the hardware have been installed, BNU must be configured before users on the system can begin to schedule BNU jobs.

BNU is part of the AIX Optional Program Products. To install it, follow the steps outlined in How to Install AIX for RISC System/6000 Licensed Program Optional Program Products in *General Concepts and Procedures*.

For information about setting up BNU after it has been installed on your system, read Configuring BNU on page 6–13.

Related Information

TCP/IP Overview for System Management on page 14–1 discusses setting up a TCP/IP network.

How to Install AIX for RISC System/6000 Licensed Program Optional Program Products in *General Concepts and Procedures*.

The **smit** command starts SMIT; Devices in SMIT in the SMIT Overview in *General Concepts and Procedures* discusses using SMIT to configure modems and communication ports.

Configuring BNU

In order for BNU to function correctly at your site, you must configure the remote communications facilities by setting up the following:

- A list of the devices used to establish a hardwired communications link or a link using a telephone line and a modem
- A list of modems used to contact remote systems over the telephone network
- A list of the remote systems with which the local system can communicate
- An optional list of alphabetic abbreviations representing the prefixes of telephone numbers used to contact the specified remote systems
- The appropriate access permissions specifying the ways in which local and remote systems may communicate
- A schedule for monitoring the networked remote systems (optional).

To set up these lists, permissions, schedules, and procedures, modify the BNU Configuration files and edit the **/usr/spool/cron/crontabs/uucp** file to remove the comment characters (#) from the beginnings of the lines that schedule the automatic maintenance routines.

It is not necessary to modify the BNU configuration files in any particular order. However, the **Systems**, **Devices**, and **Permissions** files must be configured before BNU will run correctly at your site.

For further information about configuring BNU, read How to Configure BNU on page 6–23 and How to Configure BNU for use with TCP/IP on page 6–26.

For further information about the files used to configure BNU, read List of BNU Configuration Files on page 6–21 and Understanding the BNU Configuration Files on page 6–8. For a checklist of information you need to set up BNU on your system, read List of Information to Collect Before Configuring BNU on page 6–25.

Installing BNU on page 6–12 discusses the hardware and software you need to use BNU. Maintaining BNU on page 6–14 discusses monitoring BNU communications after BNU is installed.

Related Information

Example of a BNU Configuration for a Telephone Connection on page 6–37, Example of a BNU Configuration for a TCP/IP Connection on page 6–34, and Example of a BNU Configuration for a Direct Connection on page 6–40 show sample entries in all of the BNU configuration files.

How to Monitor a BNU Remote Connection on page 6–29 and How to Use the uucico Daemon to Debug BNU Login Failures on page 6–32 can help you to check your configuration to be sure it is working correctly.

Maintaining BNU

BNU must be maintained so that it continues to work properly on your system. To maintain BNU, you should:

- Read and remove log files periodically.
- Use the **uuq** and **uustat** commands to check the BNU queues to be sure jobs are transferring to remote systems properly.
- Schedule automatic commands which poll remote systems for jobs, return unsent files to users, and send you periodic messages about BNU status.
- Periodically update the configuration files to reflect changes in your system.

In addition, you should occasionally check with administrators of remote systems to keep up with changes on their systems that may affect your configuration. For example, if the supervisor of system `venus` changes your system's password, you will need to put the new password in the **/usr/lib/uucp/Systems** file before your system can log in to system `venus`.

For further information about log files, read Working with BNU Log Files on page 6–15. For further information about the commands available for maintaining BNU, read Using BNU Maintenance Commands on page 6–16.

List of BNU Management Commands on page 6–18 contains a list of commands that can be used to maintain BNU.

For further information about BNU directories, read Understanding the BNU Administrative Directories and Files on page 6–9.

Related Information

How to Set Up Automatic Monitoring of BNU on page 6–27, How to Set Up BNU Polling of Remote Systems on page 6–28.

How to Monitor a BNU Remote Connection on page 6–29, How to Monitor a BNU File Transfer on page 6–31, and How to Use the uucico Daemon to Debug BNU Login Failures on page 6–32 can help you diagnose and correct problems with BNU connections.

Working with BNU Log Files

BNU creates log files and error files to track its own activities. These files must be checked and removed periodically to keep them from filling up the storage space on your system. BNU provides several commands for use in cleaning up log files: the **uulog** command, **uuclean** command, **uucleanup** command, and **uudemon.cleanu** command. You can run these commands manually or use entries in the **/usr/spool/cron/crontabs/uucp** file to schedule them to be run by the **cron** daemon.

Log Files in the .Log and .Old Directories

BNU creates individual log files in the **/usr/spool/uucp/.Log** directory for each remote system with which your local system communicates using the **uucp**, **uuto**, or **uux** command. BNU normally places status information about each transaction in the appropriate log file each time someone on the system uses BNU. When more than one BNU process is running, however, the system cannot access the log file, so it places the status information in a separate file with a **.LOG** prefix.

The **uulog** command displays a summary of **uucp** or **uux** requests by user or by system. The **uulog** command simply displays the files. However, you can also have BNU automatically combine the log files into a primary log file. This is called *compacting* the log files, and can be done with the **uudemon.cleanu** command, which is usually run by the **cron** daemon.

Whenever the **cron** daemon runs the **uudemon.cleanu** command, that command combines the **uucico** and **uuxqt** log files on the local system and stores them in the **/usr/spool/uucp/.Old** directory. At the same time, the command removes old log files that were previously stored in the **.Old** directory. The default is for the **uudemon.cleanu** command to save log files that are two days old.

If storage space is a problem, consider reducing the number of days that files are kept. If you need to be able to track BNU transactions over a longer period of time, consider increasing the number of days that files are kept. To change the default amount of time to save log files, modify the shell procedure for the **uudemon.cleanu** command. This script is stored in the **/usr/lib/uucp** directory and can be modified by someone with root user authority.

Other BNU Log Files

BNU also collects information and stores it in the **/usr/spool/uucp/.Admin** directory. This directory contains the **errors**, **xferstats**, **Foreign**, and **audit** files. These files must be checked and removed occasionally to save storage space. BNU re-creates each file whenever it is needed.

When another system contacts your system with the **uucico** daemon's debugging mode on, it invokes the **uucico** daemon on your system with debugging turned on as well. The debugging messages generated by the daemon on the local system are stored in the **audit** file. This file can get quite large if remote users frequently call your system with debugging turned on. Check and remove the **audit** file often.

The **errors** file records errors encountered by the **uucico** daemon. Checking this file can help you correct problems such as incorrect permissions on BNU work files.

The **xferstats** file contains information about the status of every file transfer. Check and remove this file occasionally.

The **Foreign** file is important to the security of your system. Whenever an unknown system attempts to log in to the local system, BNU calls the **remote.unknown** shell procedure, which logs the attempt in the **Foreign** file. By reading the **Foreign** file, you can find out which systems have attempted to call the local system and been refused. If a system has been attempting frequent calls, you can use this information when considering whether to allow that system access.

Systemwide Log Files used by BNU

Because many BNU processes need root user authority to complete their tasks, BNU creates frequent entries in the **/usr/spool/sulog** log file. Similarly, using the **cron** daemon to schedule BNU tasks causes BNU to create multiple entries in the **/usr/spool/cron/log** file. When using BNU, check and clean out these files more often than you otherwise would.

For further information about BNU log files, read Understanding the BNU Administrative Directories and Files on page 6–9.

Related Information

The **crontab** command, **uuclean** command, **uucleanup** command, **uudemon.cleanu** command, **uulog** command.

The **/usr/spool/uucp/.Log/*** directories, **/usr/spool/uucp/.Old/*** directories.

Using BNU Maintenance Commands

The Basic Networking Utilities contain several commands that can be used for monitoring BNU activities and cleaning up BNU directories and files.

Cleanup Commands

BNU contains three commands that clean directories and remove any files that have not yet been sent. These commands are the **uuclean**, **uucleanup**, and **uudemon.cleanu** commands.

The **uuclean** command allows a user with root user authority to delete from the BNU administrative directories all files older than a specified number of hours. Using the **uuclean** command, you can also specify a certain directory to be cleaned or a certain type of file to be deleted. In addition, you can instruct the command to notify the owners of the files that their files have been deleted. The **uuclean** command is the Berkeley equivalent of the **uucleanup** command.

The **uucleanup** command performs functions similar to the **uuclean** command. However, the **uucleanup** command checks the age of files based on *days* rather than hours. In addition, you can use the **uucleanup** command to send a warning message to users whose files have not yet been transferred, notifying them that the files are still in the queue. The **uucleanup** command also allows you to remove only files relating to a specified remote system.

The **uudemon.cleanu** command is a shell procedure that issues the **uulog** and **uucleanup** commands to compress the BNU log files and to remove log and work files over three days old. It is usually executed by the **cron** daemon.

Status-checking Commands

BNU also provides several commands for checking the status of transfers and log files. These are the **uuq**, **uulog**, **uupoll**, **uusnap**, and **uustat** commands.

The **uuq** command displays jobs currently in the BNU job queue. Using the **uuq** command, you can display the status of a specified job or of all jobs. If you have root user authority, you can use the **uuq** command to delete a job from the queue.

The **uustat** command provides information similar to that provided by the **uuq** command, but in a different format. Using **uustat**, you can check the status of jobs and delete jobs you own. If you have root user authority, you can also delete jobs belonging to other users.

The **uulog** command works with BNU log files (see page 6–15).

The **uupoll** command forces a poll of a remote system. This is helpful when work for that system is waiting in the queue and needs to be transferred before the system is scheduled to be called automatically.

The **uusnap** command displays a very brief summary of BNU status. For each remote system, this command shows the number of files awaiting transfer. However, it does not show how long they have been waiting. The **uusnap** command is the Berkeley equivalent of the **uustat** command.

Shell Procedures

BNU is delivered with two shell procedures that can be used for maintenance: the **uudemon.cleanu** command and the **uudemon.admin** command. They are stored in the **/usr/lib/uucp** directory. Since these commands are shell procedures, you can copy them and modify the copy if you wish to change what they do. You can run the procedures from the command line or schedule them to be run by the **cron** daemon.

The **uudemon.cleanu** command is discussed under **Cleanup Commands**.

The **uudemon.admin** command issues the **uustat** command to find out the status of BNU jobs. It sends the results to the **uucp** login ID as mail. You may wish to modify the **uudemon.admin** shell procedure to send the mail elsewhere, or use a mail program to reroute all mail for the **uucp** login ID to the user who is responsible for BNU administration.

To automatically run the **uudemon.cleanu** and **uudemon.admin** commands, remove the comment characters (#) from the beginnings of the relevant lines in the **/usr/spool/cron/crontabs/uucp** file.

Related Information

The **uuclean** command, **uucleanup** command, **uudemon.admin** command, **uudemon.cleanu** command, **uupoll** command, **uuq** command, **uusnap** command.

The **cron** daemon.

List of Information for BNU System Management

The reference information about managing BNU falls into the following general categories:

Commands are listed beginning on page 6–18.

Daemons are listed on page 6–20.

Files, File Formats, and Directories are listed together beginning on page 6–20.

Directories are listed on page 6–21.
File Formats are listed on page 6–21.
Files are listed on page 6–22.

List of BNU Management Commands

BNU management commands fall into four general categories:

Installation Commands

Configuration Command

Maintenance and Debugging Commands

Automatic Maintenance Commands.

In addition, you may wish to become familiar with the BNU user commands, which are listed on page 6–19.

Installation Commands

Cvt	Moves UUCP files from a non-BNU version of UUCP into new BNU directories.
uucheck	Checks the **/usr/lib/uucp/Permissions** file for correct configuration.
uuname	Shows the names of all the systems BNU can connect to.

Configuration Command

uucpadm	Enters basic BNU configuration information.

Maintenance and Debugging Commands

uuclean	Cleans up BNU spooling directories.
uucleanup	Cleans up BNU spooling directories.
uucpadm	Enters basic BNU configuration information.
uukick	Contacts a remote system with debugging turned on.
uulog	Displays BNU log files.
uupoll	Forces a poll of a remote system.
uuq	Displays the BNU job queue and allows specified jobs to be deleted from the queue.
uusnap	Displays a brief summary, like a snapshot, of the status of BNU.
uustat	Provides information about BNU operations.
uutry	Contacts a remote system with debugging turned on and allows an override of the retry time.
Uutry	Contacts a remote system with debugging turned on, saving the debugging output in a file.

Automatic Maintenance Commands

Although these commands can be run from the command line, they are usually run by the **cron** daemon at times determined by the person who configures BNU.

uudemon.admin	Reports BNU status by sending mail to a specified login ID, by default the **uucp** login ID.
uudemon.cleanu	Cleans up BNU spooling directories at prescheduled times.
uudemon.hour	Initiates file transport calls to remote systems.
uudemon.poll	Polls remote systems listed in the **/usr/lib/uucp/Poll** file.

Related Information

The commands for managing BNU are described in the *Commands Reference*.

Installing BNU on page 6–12 discusses the use of the installation commands.

Configuring BNU on page 6–13 discusses setting up the automatic maintenance commands.

Working with BNU Log Files on page 6–15 and Using BNU Maintenance Commands on page 6–16 provide information about using the maintenance commands.

List of BNU User Commands

Once BNU is installed and configured, the following commands can be run by any AIX user to cause BNU to log in to remote systems, transfer files, perform jobs on remote systems, and report the status of jobs and transfers.

ct	Dials a remote system and initiates a login process.
cu	Connects directly or indirectly to another system.
uucp	Copies files to another AIX or UNIX system that is running BNU or another version of UUCP.
uudecode	Decodes a binary file that was encoded by the **uuencode** command.
uuencode	Encodes a binary file for transmission by mail.
uuname	Provides information about other systems accessible to the local system.
uupoll	Forces a call to a remote system so queued jobs can be transferred.
uuq	Displays the BNU job queue.
uusend	Uses BNU to send a file to a remote host that is running BNU or another version of UUCP.
uusnap	Displays a brief summary, like a snapshot, of the status of BNU.
uustat	Reports the status of and provides limited control over BNU operations.
uuto	Copies files to another system that is using BNU or another version of UUCP.
uupick	Completes the transfer of and handles files sent by the **uuto** command.
uux	Runs a command on another AIX or UNIX system that is running BNU or another version of UUCP.

Related Information

The BNU user commands are described in the *Commands Reference*.

List of BNU Daemons

BNU has four daemons:

uucico	Contacts remote systems and transfers files between the local system and a remote system.
uucpd	Allows BNU to run on top of TCP/IP.
uusched	Schedules BNU jobs.
uuxqt	Executes command requests from remote systems.

Related Information

The BNU daemons are described in the *Commands Reference* and in Understanding the BNU Daemons on page 6–5.

List of BNU Files and Directories

BNU files and directories fall into several logical groups:

Lock and Public Directories

Administrative Directories and Files

Configuration Files.

Lock and Public Directories

/etc/locks directory

/usr/spool/uucppublic/* directories.

Administrative Directories and Files

The BNU administrative directories and files are:

/usr/spool/uucppublic/* directories

/usr/spool/uucp directory

/usr/spool/uucp/.Admin directory

- **audit** file
- **errors** file
- **Foreign** file
- **xferstats** file

/usr/spool/uucp/.Corrupt directory

/usr/spool/uucp/.Log directories

/usr/spool/uucp/.Old directory

/usr/spool/uucp/.Status directory

/usr/spool/uucp/.Workspace directory

/usr/spool/uucp/.Xqtdir directory

/usr/spool/uucp/*SystemName* directory

- Command (**C.***) files
- Data (**D.***) files
- Execute (**X.***) files
- Temporary (**TM.***) files.

Configuration Files

The BNU configuration files, which reside in the **/usr/lib/uucp** directory, are:

Devices file format

Dialers file format

Dialcodes file format

Maxuuscheds file format

Maxuuxqts file format

Permissions file format

Poll file format

remote.unknown file format

Systems file format.

Related Information

For information about individual files, file formats, and directories, see the *Files Reference*.

Understanding the BNU File and Directory Structure on page 6–7, Understanding the BNU Configuration Files on page 6–8, and Understanding the BNU Administrative Directories and Files on page 6–9 discuss the BNU files and directories.

List of BNU Directories

BNU uses the following directories:

/etc/locks

/usr/spool/uucp

/usr/spool/uucp/.Admin

/usr/spool/uucp/.Corrupt

/usr/spool/uucp/.Log

/usr/spool/uucp/.Old

/usr/spool/uucp/.Status

/usr/spool/uucp/.Workspace

/usr/spool/uucp/.Xqtdir

/usr/spool/uucp/*SystemName*

/usr/spool/uucppublic/*

Related Information

For information about individual directories, see the *Files Reference*.

Understanding the BNU File and Directory Structure on page 6–7 and Understanding the BNU Administrative Directories and Files on page 6–9 discuss the BNU directories.

List of BNU File Formats

The configuration files for BNU are located in the **/usr/lib/uucp** directory:

Devices

Dialers

Dialcodes

Maxuuscheds

Maxuuxqts

Permissions

Poll

remote.unknown

Systems

Related Information

For information about individual file formats, see the articles on each in the *Files Reference*.

Understanding the BNU File and Directory Structure on page 6–7 and Understanding the BNU Configuration Files on page 6–8 discuss the BNU file formats.

List of BNU Files

Files in the **/usr/spool/uucp/.Admin** directory are:

audit

errors

Foreign

xferstats

Files in the **/usr/spool/uucp/***SystemName* directory include:

Command (**C.***)

Data (**D.***)

Execute (**X.***)

Temporary (**TM.***)

Related Information

For information about individual files, see the *Files Reference*.

Understanding the BNU File and Directory Structure on page 6–7 and Understanding the BNU Administrative Directories and Files on page 6–9 discuss the BNU files.

How to Configure BNU

Prerequisite Tasks or Conditions

- BNU/UUCP must be installed on your system.
- You must have root user authority to edit the BNU configuration files.
- Collect the information you need to configure BNU. This information should include a list of remote systems and lists of devices and modems to use to connect to the systems.
- If you are using direct connections for BNU communications, the appropriate hardwired connections between your system and the remote systems must be set up.
- If you are using modems for BNU communications, you must have installed and configured each modem.
- If one or more of your connections uses TCP/IP, then TCP/IP must be running between your system and the appropriate remote systems. Follow the steps described in How to Configure BNU for Use with TCP/IP on page 6–26 before continuing the configuration.

Procedure

To configure BNU on your system, perform the following steps:

1. Set up appropriate login IDs and passwords for remote systems that will call your system, and tell the person responsible for administering BNU or UUCP on each remote system the login and password you have provided.

   Warning: Allowing remote systems to log into the local system with the **uucp** login ID seriously jeopardizes the security of your system. Remote systems logged in with the **uucp** ID can display and possibly modify (depending on the permissions specified in the LOGNAME entry of the **Permissions** file) the local **Systems** and **Permissions** files. It is strongly recommended that you create other BNU login IDs for remote systems and reserve the **uucp** login ID for the person administering BNU on the local system. For the best security, each remote system that contacts the local system should a have unique login ID with a unique UID number. These login IDs should have GIDs of 5.

2. Decide whether to use dial-code abbreviations for telephone numbers (see the **Dialcodes** file format). If you decide to use dial-code abbreviations, set up the **Dialcodes** entry for each abbreviation.
3. Using your list of devices, set up the **Devices** file on your system. Make an entry for each modem and each direct connection. If you are using TCP/IP, make sure the TCP/IP entry is included in the **Devices** file.
4. Using your information about each remote system, set up the **Systems** file on your system.
5. Using your information about devices and modems, make sure the **Dialers** file on your system contains an entry for each modem. If you are using TCP/IP and direct connections, make sure the TCP/IP entry and direct entries are present.
6. Decide how much access to your system you want to provide to each remote system you call and to each remote system that calls you. Set up appropriate entries for each system and each login name in the **Permissions** file.
7. Issue the **uucheck** command to verify that everything is in place:

   ```
   /usr/lib/uucp/uucheck -v
   ```

The **uucheck** command verifies that the directories, programs, and support files are set up properly and that the **Permissions** file entries are consistent. If the **uucheck** command reports any errors, fix the errors.

8. If you wish, set up automatic monitoring of BNU operations and automatic polling of remote systems.

Related Information

BNU/UUCP is installed with the AIX Optional Program Products. See How to Install AIX for RISC System/6000 Licensed Program Optional Program Products in *General Concepts and Procedures*.

List of Information to Collect Before Configuring BNU on page 6–25.

All BNU configuration files are located in the **/usr/lib/uucp** directory. The formats for the configuration files are the **Devices** file format, **Dialers** file format, **Dialcodes** file format, **Maxuuscheds** file format, **Maxuuxqts** file format, **Permissions** file format, **Poll** file format, **remote.unknown** file format, and **Systems** file format.

The **uucpadm** command provides menu-driven configuration for the **Devices**, **Dialcodes**, **Permissions**, **Poll**, and **Systems** files, which can also be edited using any ASCII editor. The other configuration files must be edited with an ASCII editor.

The **uucheck** command verifies the BNU configuration.

How to Configure BNU for Use with TCP/IP on page 6–26, How to Set Up Automatic Monitoring of BNU on page 6–27, How to Set Up BNU Polling of Remote Systems on page 6–28.

How to Monitor a BNU Remote Connection on page 6–29 and How to Use the uucico Daemon to Debug BNU Login Failures on page 6–32 can help you locate problems in the configuration.

List of Information to Collect before Configuring BNU

Before beginning to configure BNU, gather the information listed here.

- For each remote system your system will call, collect the following information:
  - System name
  - Login name your system should use on the remote system
  - Password for the login name
  - Login and password prompts on the remote system
  - Type of connection you will use to reach the remote system (TCP/IP, direct, or telephone).
- If the connection is direct, find out:
  - The bit rate of the connection
  - The port on the local system to which the connection is attached.
- If the connection is a telephone connection, find out:
  - The telephone number of the remote system
  - The speed of your modem that is compatible with that of the remote system.

  Note: If any of the remote systems will call your system, make sure the person responsible for administering BNU on each of the remote systems has all the preceding information about your system.
- For each local modem that you will use for BNU connections, collect the following information:
  - The chat script for the modem (consult the modem's documentation)

    Note: For some modems, the chat script is already in the **/usr/lib/uucp/Dialers** file.
  - The local port the modem is attached to.
- Using the information you collect, make a list of each device you need to connect to a remote system. Following is a sample list for local system `morgan`:

```
direct:
hera 9600 tty5
zeus 2400 tty2
ariadne 2400 tty1

hayes modem (tty3):  apollo, athena

TCP/IP:  merlin, arthur, percy
```

To connect to system `hera`, a `direct` connection at a speed of `9600` from port `tty5` is used. To connect to system `apollo`, the `hayes` modem connected to port `tty3` is used. TCP/IP is used to connect to systems `merlin`, `arthur`, and `percy`.

Related Information

How to Configure BNU on page 6–23, How to Configure BNU for Use with TCP/IP on page 6–26.

How to Configure BNU for Use with TCP/IP

Prerequisite Tasks or Conditions

- BNU/UUCP must be installed on your system.
- You must have root user authority to edit the BNU and TCP/IP configuration files.
- TCP/IP must be running between your system and the appropriate remote systems.

Procedure

1. Use the **netstat** command to see whether the **uucpd** daemon is running, by entering:

   ```
   netstat -a
   ```

   The **uucpd** daemon is started by the **inetd** daemon. If the **uucpd** daemon is not running, reconfigure the **inetd** daemon to start the **uucpd** daemon.

2. Check to see whether the **/etc/services** file includes the following line:

   ```
   uucp            540/tcp         uucpd
   ```

   If not, add the line.

3. Check the host-name table in the **/etc/hosts** file to be sure that it includes the name of the remote computer with which you want to connect.
4. Update the **Devices** and **Dialers** files in the **/usr/lib/uucp** directory to include the TCP/IP entries.
5. Complete the configuration for each remote system and for your system as described in How to Configure BNU.

Related Information

The **/usr/lib/uucp/Devices** file format, **/usr/lib/uucp/Dialers** file format.

The **uucpadm** command can be used to make entries in the **Devices** file.

BNU/UUCP is installed with the AIX Optional Program Products. See How to Install AIX for RISC System/6000 Licensed Program Optional Program Products in *General Concepts and Procedures*.

How to Configure BNU on page 6–23.

The **netstat** command.

The **uucpd** daemon, **inetd** daemon.

Understanding Installation and Configuration for TCP/IP on page 14–80 discusses setting up a TCP/IP network.

How to Configure the inetd Daemon on page 14–102.

The **/etc/hosts** file format, **/etc/inetd.conf** file format, **/etc/services** file format.

How to Set Up Automatic Monitoring of BNU

Prerequisite Tasks or Conditions

- Complete the steps in How to Configure BNU on page 6–23 before beginning this procedure.
- You must have root user authority to edit the **/usr/spool/cron/crontabs/uucp** file.

Procedure

BNU uses the **cron** daemon to start BNU daemons and to monitor BNU activity. The **cron** daemon reads the **/usr/spool/cron/crontabs/uucp** file for instructions about when to start BNU procedures.

1. Log in as a user with root user authority.
2. Using an ASCII text editor, edit the **/usr/spool/cron/crontabs/uucp** file.
3. Uncomment the lines for the BNU maintenance procedures, **uudemon.admin** and **uudemon.cleanu**. You can change the times these procedures are run if your system needs maintenance at more or less frequent intervals. It is best, however, to run the **uudemon.admin** command at least once a day and the **uudemon.cleanu** command at least once a week.
4. You can use the **crontabs/uucp** file to schedule other BNU maintenance commands, such as the **uulog**, **uuclean**, or **uucleanup** commands. In addition, you can use the **crontabs/uucp** file to instruct the **cron** daemon to start the **uucico**, **uuxqt**, or **uusched** daemons at specific times.

Related Information

How to Configure BNU on page 6–23.

The **cron** daemon runs the commands that monitor BNU.

The **uudemon.admin** command reports BNU status to a designated login ID (by default, the **uucp** login ID). The **uudemon.cleanu** command cleans up BNU spooling directories and compacts the BNU log files.

The **uuclean** and **uucleanup** command clean up BNU spooling directories. The **uulog** command displays BNU log files.

The **uucico** daemon contacts remote systems and transfers files between the local system and a remote system. The **uusched** daemon schedules BNU jobs and calls the **uucico** daemon. The **uuxqt** daemon executes command requests from remote systems.

How to Set Up BNU Polling of Remote Systems

Prerequisite Tasks or Conditions

- Complete the steps in How to Configure BNU on page 6–23 before beginning this procedure.
- You must have root user authority to edit the **/usr/spool/cron/crontabs/uucp** file and the **/usr/lib/uucp/Poll** file.

Procedure

So that BNU will poll remote systems for jobs, the systems must be listed in the **/usr/lib/uucp/Poll** file, and the **uudemon.hour** and **uudemon.poll** commands must be run periodically.

1. Decide which remote systems you want to poll automatically and how often you want to poll each one. The **Poll** file allows you to specify times for each system: as seldom as once a day or as often during the day as you wish.
2. Log in as a user with root user authority.
3. Using an ASCII text editor or the **uucpadm** command, edit the **Poll** file and add an entry for each system your system will poll.

   Note: The systems listed in the **Poll** file must also be listed in the **/usr/lib/uucp/Systems** file.

4. Using an ASCII text editor, edit the **/usr/spool/cron/crontabs/uucp** file. Remove the comment characters (#) from the beginnings of the lines in the that run the **uudemon.hour** and **uudemon.poll** commands. You can change the times these commands are run. If you choose to do this, however, be sure to schedule the **uudemon.poll** command approximately five minutes *before* you schedule the **uudemon.hour** command.

BNU will now automatically poll the systems listed in the **Poll** file at the times you have specified.

Related Information

How to Configure BNU on page 6–23.

The **cron** daemon runs the commands that poll remote systems.

The **/usr/lib/uucp/Poll** file format, **/usr/lib/uucp/Systems** file format.

The **uudemon.hour** command and **uudemon.poll** command poll remote systems.

How to Monitor a BNU Remote Connection

Prerequisite Tasks or Conditions

- The BNU program must be installed on your system.
- A link (hardwired, modem, or TCP/IP) must be set up between your system and the remote system.
- The BNU configuration files, including the **Systems** file, **Permissions** file, **Devices** file, and **Dialers** file, must be set up for communications between your system and the remote system.

  Note: You must have root user authority to modify the BNU configuration files.

Procedure

The **Uutry** command can help you monitor the **uucico** daemon process if users at your site report file–transfer problems.

1. Issue the **uustat** command to determine the status of all the transfer jobs in the current queue as follows:

   ```
   uustat –q
   ```

   The system displays a status report like the following:

   ```
   venus 3C (2) 05/09–11:02 CAN'T ACCESS DEVICE
   hera 1C 05/09–11:12 SUCCESSFUL
   merlin 2C 5/09–10:54 NO DEVICES AVAILABLE
   ```

   This report indicates that three command (**C.***) files intended for remote system `venus` have been in the queue for two days. There could be several reasons for this delay. For example, perhaps system `venus` has been shut down for maintenance or the modem has been turned off.

2. Before you begin more extensive troubleshooting activities, issue the **Uutry** command as follows to determine whether your local system can contact system `venus` at this time:

   ```
   /usr/lib/uucp/Uutry –r venus
   ```

   This command starts the **uucico** daemon with a moderate amount of debugging and the instruction to override the default retry time. The **Uutry** command directs the debugging output to a temporary file, `/tmp/venus`.

3. If your local system succeeds in establishing a connection to system `venus`, the debugging output contains a good deal of information. However, the final line in this script, which follows, is the most important:

   ```
   Conversation Complete: Status SUCCEEDED
   ```

   If the connection is successful, assume that the temporary file–transfer problems are resolved. Issue the **uustat** command again to make certain that the files in the spooling directory have been transferred successfully to the remote system. If they have not, use the steps in How to Monitor a BNU File Transfer on page 6–31 to check for file–transfer problems between your system and the remote system.

4. If your local system cannot contact the remote system, the debugging output generated by the **Uutry** command contains the following type of information (the exact form of the output may vary):

```
mchFind called (venus)
conn (venus)
getto ret -1
Call Failed: CAN'T ACCESS DEVICE
exit code 101
Conversation Complete: Status FAILED
```

   First, check the physical connections between the local and remote systems. Make sure that the remote computer is turned on and all cables are properly connected, that the ports are enabled or disabled (as appropriate) on both systems, and that the modems (if applicable) are working.

   If the physical connections are correct and secure, then verify all the relevant configuration files on both the local and remote systems, including the following:

   - Make certain that the entries in the **Devices**, **Systems**, and **Permissions** files (in the **/usr/lib/uucp** directory) are correct on both systems.
   - If you are using a modem, make sure that the **/usr/lib/uucp/Dialers** file contains the proper entry. If you are using dial-code abbreviations, be sure the abbreviations are defined in the **/usr/lib/uucp/Dialcodes** file.
   - If you are using a TCP/IP connection, make sure that the **uucpd** daemon is running on both the local system and the remote system and that the configuration files contain the correct TCP entries.

5. Once you have checked the physical connections and configuration files, issue the **Uutry** command again. If the debugging output still reports that the connection failed, you may need to confer with a member of your systems support team. Save the debugging output produced by the **Uutry** command. This may prove helpful in diagnosing the problem.

Related Information

The **uucico** daemon and **uucpd** daemon.

The **uucp** command, **uustat** command, **Uutry** command.

How to Use the uucico Daemon to Debug BNU Login Failures on page 6–32, How to Monitor a BNU File Transfer on page 6–31.

How to Configure BNU on page 6–23, How to Configure BNU for Use with TCP/IP on page 6–26.

How to Monitor a BNU File Transfer

Prerequisite Tasks or Conditions

1. The BNU program must be installed on and configured for your system.
2. Establish a connection to a remote system using the steps given in How to Monitor a BNU Remote Connection on page 6–29.

Monitoring a File Transfer

Use this procedure to monitor a file transfer to a remote system. Monitoring a file transfer is useful when file transfers to the remote system in question are failing for unknown reasons. The debugging information produced by the **uucico** daemon (called by the **Uutry** command) can help you find out what is working incorrectly.

The **Uutry** command enables you to monitor file transfers, as follows:

1. Prepare a file for transfer using the **uucp** command with the **–r** flag, by entering:

   ```
   uucp –r test1 venus!~/test2
   ```

   The **–r** flag instructs the BNU program to place the `test1` file in the queue, but *not* to start the **uucico** daemon.

2. Issue the **Uutry** command with the **–r** flag to start the **uucico** daemon with debugging turned on, by entering:

   ```
   /usr/lib/uucp/Uutry –r venus
   ```

   This instructs the **uucico** daemon to contact remote system `venus` overriding the default retry time. The daemon contacts system `venus`, logs in, and transfers the file, while the **Uutry** command produces debugging output that enables you to monitor the **uucico** process. Press the Interrupt key sequence to stop the debugging output and return to the command prompt.

   The **Uutry** command also stores the debugging output in the **/tmp/***SystemName* file. If you break out of the debugging output before the connection is complete, you can page the output file to see the outcome of the connection.

Related Information

The **Uutry** command calls the **uucico** daemon to transfer the file.

The **uucp** command, **uustat** command, **Uutry** command.

The default retry time is specified in a file named for the remote system and placed in the **/usr/spool/uucp/.Status** directory.

How to Monitor a BNU Remote Connection on page 6–29 explains how to establish a connection to a remote system.

How to Configure BNU on page 6–23, How to Configure BNU for Use with TCP/IP on page 6–26.

How to Use the uucico Daemon to Debug BNU Login Failures

Prerequisite Tasks or Conditions

- BNU must be installed on your system.
- A link (hardwired, modem, or TCP/IP) must be set up between your system and the remote system.
- The BNU configuration files, including the **Systems** file, **Permissions** file, **Devices** file, and **Dialers** file, must be set up for communications between your system and the remote system.

  Note: You must have root user authority to modify the BNU configuration files.
- You must have root user authority to invoke the **uucico** daemon in debugging mode.

Procedure

1. To produce debugging information about a local-to-remote system connection that is not working, start the **uucico** daemon with the **–x** flag as follows:

   ```
   /usr/lib/uucp/uucico –r 1 –s venus –x 9
   ```

   where `–r 1` specifies the server mode; `–s venus`, the name of the remote system to which you are trying to connect; and `–x 9`, the debug level that produces the most detailed debugging information.

2. If the expect-send sequence entry in the **/usr/lib/uucp/Systems** file is:

   ```
   venus Any venus 1200 – "" \n in:—in: uucp1 word: mirror
   ```

   the debugging output produced when the **uucico** daemon connects the local system to the remote system `venus` is similar to:

   ```
   expect: ""
   got it
   sendthem (^J^M)
   expect (in:)^
   M^Jlogin:got it
   sendthem (uucp1^M)
   expect (word:)^
   M^JPassword:got it
   sendthem (mirror^M)
   imsg >^M^J^PShere^@Login Successful: System=venus
   ```

   These messages represent the following information:

   `expect: ""`
   : The local system should not wait for any information from the remote system.

   `got it`
   : Acknowledgment.

   `sendthem (^J^M)`
   : The local system sends the remote system a carriage return and a new line.

   `expect (in:)`
   : The local system expects to receive the remote system login prompt, which ends in the `in:` character string.

`^M^Jlogin:got it`
: The local system receives the remote login prompt.

`sendthem (uucp1^M)`
: The local system sends the `uucp1` login ID to the remote system.

`expect (word:)`
: The local system expects to receive the remote system password prompt, which ends in the `word:` character string.

`^M^JPassword:got it`
: The local system receives the remote password prompt.

`sendthem (mirror^M)`
: The local system sends the password for the `uucp1` login ID to the remote system.

`imsg >^M^J^PShere^@Login Successful: System=venus`
: The local system is successfully logged in to remote system `venus`.

Notes:

1. The expect-send debugging output produced by the **uucico** command can come either from information in the **/usr/lib/uucp/Dialers** file or from information in the **/usr/lib/uucp/Systems** file. Information about communication with the modem comes from the **Dialers** file, while information about communication with the remote system comes from the **Systems** file.
2. To set up a connection with a remote system, you must be familiar with the login sequence of that system.

Related Information

How to Configure BNU on page 6–23, How to Configure BNU for Use with TCP/IP on page 6–26, How to Monitor a BNU Remote Connection on page 6–29, How to Monitor a BNU File Transfer on page 6–31.

The **ct** command and **cu** command can be used to establish BNU remote connections.

The **Devices** file format, **Dialers** file format, **Permissions** file format, and **Systems** file format, located in the **/usr/lib/uucp** directory.

Example of a BNU Configuration for a TCP/IP Connection

The following files are set up for a TCP/IP connection between systems zeus and hera, where zeus is considered the local system and hera the remote system.

Entries in the Local System's Files

The **Systems** file on system zeus contains the following entry allowing zeus to contact system hera:

```
hera  Any  TCP,t  -  -  in:—in: uzeus  word:  birthday
```

This specifies that system zeus can call system hera at any time, using the t protocol for communications with system hera. System zeus logs in to system hera as uzeus with the password birthday.

Note: The t protocol supports the **tcp** protocol. Therefore, always use the t protocol for BNU communications over TCP/IP connections.

BNU uses the *Type* and *Class* fields in the **Systems** file to find the appropriate device for the connection. Accordingly, it checks the **Devices** file for an entry of type TCP.

The **Devices** file on system zeus contains the following entry for TCP/IP connections:

```
TCP  -  -  -  TCP
```

Because the device type is TCP, there are no *Class*, *Line*, or *Line2* entries. The *Dialer* is also specified as TCP. Accordingly, BNU looks in the **Dialers** file for a TCP entry.

The **Dialers** file on system zeus contains a TCP/IP entry as follows:

```
TCP
```

This entry specifies that no handshaking is required.

Note: Handshaking is never required over a TCP/IP connection.

The **Permissions** file on system zeus contains the following entry specifying system hera's access to system zeus:

```
LOGNAME=uhera VALIDATE=hera SENDFILES=yes REQUEST=yes \
MACHINE=hera READ=/usr/spool/uucppublic:/u/hera \
WRITE=/usr/spool/uucppublic:/u/hera COMMANDS=ALL
```

This combined LOGNAME and MACHINE entry provides the following permissions to system hera on system zeus:

- System hera can request and send files regardless of who initiated the call.
- System hera can read and write to the public directory and the **/u/hera** directory on system zeus.
- System hera can execute all commands on system zeus.
- System hera must log in to system zeus as user uhera and cannot use any other login ID for BNU transactions.

Note: Because the permissions are the same regardless of which system initiates the call, the preceding LOGNAME and MACHINE entries are combined. Separately, they are:

```
LOGNAME=uhera VALIDATE=hera SENDFILES=yes REQUEST=yes \
READ=/usr/spool/uucppublic:/u/hera \
WRITE=/usr/spool/uucppublic:/u/hera

MACHINE=hera REQUEST=yes COMMANDS=ALL\
READ=/usr/spool/uucppublic:/u/hera \
WRITE=/usr/spool/uucppublic:/u/hera
```

Entries in the Remote System's Files

The **Systems** file on system `hera` contains the following entry allowing `hera` to contact system `zeus`:

```
zeus  Any  TCP,t  -  -  ogin:—ogin: uhera  ord: lightning
```

This specifies that system `hera` can call system `zeus` at any time, using the **t** protocol for communications with system `zeus`. System `hera` logs in to system `zeus` as user `uhera` with the password `lightning`. Again, BNU next checks the **Devices** file for an entry of type `TCP`.

Note: The **t** protocol supports the **tcp** protocol. Therefore, always use the **t** protocol for BNU communications over TCP/IP connections.

The **Devices** file on system `hera` contains the following entry for TCP/IP connections:

```
TCP  -  -  -  TCP
```

Because the device type is `TCP`, there are no *Class*, *Line*, or *Line2* entries. The *Dialer* is also specified as `TCP`. Accordingly, BNU looks in the **Dialers** file for a `TCP` entry.

The **Dialers** file on system `hera` contains a TCP/IP entry as follows:

```
TCP
```

This entry specifies that no handshaking is required.

Note: Handshaking is never required over a TCP/IP connection.

The **Permissions** file on system `hera` contains the following entry specifying system `zeus`'s access to system `hera`:

```
LOGNAME=uzeus VALIDATE=zeus SENDFILES=yes REQUEST=yes \
MACHINE=zeus COMMANDS=rmail:who:uucp
```

This combined LOGNAME and MACHINE entry provides the following permissions to system `zeus` on system `hera`:

- System `zeus` can request and send files regardless of who initiated the call.
- System `zeus` can read and write only to the public directory (the default).
- System `zeus` can execute only the **rmail**, **who**, and **uucp** commands.
- System `zeus` must log in to system `hera` as user `uzeus` and cannot use any other login ID for BNU transactions.

Note: Separately, the LOGNAME and MACHINE entries are:

```
LOGNAME=uzeus VALIDATE=zeus SENDFILES=yes REQUEST=yes

MACHINE=zeus COMMANDS=rmail:who:uucp REQUEST=yes
```

Files

/usr/lib/uucp directory	Contains all the configuration files for BNU.
/usr/lib/uucp/Devices file	Contains information about available devices.
/usr/lib/uucp/Dialers file	Specifies initial handshaking on a connection.
/usr/lib/uucp/Systems file	Describes accessible remote systems.
/usr/lib/uucp/Permissions file	Describes access permissions for remote systems.
/usr/spool/uucppublic directory	Contains files that have been transferred.

Related Information

Example of a BNU Configuration for a Telephone Connection on page 6–37, Example of a BNU Configuration for a Direct Connection on page 6–40.

How to Configure BNU on page 6–23, How to Configure BNU for Use with TCP/IP on page 6–26.

Example of a BNU Configuration for a Telephone Connection

The following sample files are set up to connect systems `venus` and `merlin` over a telephone line using modems. System `venus` is considered the local system, and system `merlin` the remote system.

On both systems, the device `tty1` is hooked to a Hayes modem at 1200 baud. The login ID for system `venus` on system `merlin` is `uvenus`, and the associated password is `mirror`. The login ID for system `merlin` on system `venus` is `umerlin`, and the associated password is `oaktree`. The phone number for the modem attached to `venus` is `9=3251436`; the number of the `merlin` modem is `9=4458784`. Both computers include partial phone numbers in their **Systems** files and dial-codes in their **Dialcodes** files.

Entries on the Local System

The **Systems** file on `venus` contains the following entry for `merlin`, including a phone number and a dialing prefix:

```
merlin Any ACU 1200 local8784 "" in:—in: uvenus word: mirror
```

System `venus` can call system `merlin` at any time, using an `ACU` device at `1200` baud and logging in as `uvenus` with the password `mirror`. The telephone number is expanded based on the code `local` in the **Dialcodes** file, and the device to be used is determined based on the *Type* and *Class* entries. Accordingly, BNU checks the **Devices** file for a device of type `ACU` and class `1200`.

The **Dialcodes** file on system `venus` contains the following dial-code prefix for use with the number in the **Systems** file:

```
local 9=445
```

Given this code, the telephone number for system `merlin` in the **Systems** file is expanded to `9=4458784`.

The **Devices** file on system `venus` contains the following entry for the connection to system `merlin`:

```
ACU tty1  -  1200  hayes \T
```

The port to be used is `tty1`, and the *Dialer* entry in the *Dialer–Token Pairs* field is `hayes`. The *Token* entry, `\T`, indicates that the telephone number is to be expanded using a code from the **Dialcodes** file. BNU checks the **Dialers** file for a `hayes` dialer.

The **Dialers** file on system `venus` contains the following entry for the `hayes` modem:

```
hayes  =,-,  ""  \dAT\r\c OK \pATDT\T\r\c CONNECT
```

Note: The expect-send characters are defined in the **Dialers** file format.

The **Permissions** file on system `venus` contains the following entries specifying the ways in which system `merlin` can conduct **uucico** and **uuxqt** transactions with system `venus`:

```
LOGNAME=umerlin VALIDATE=merlin REQUEST=yes SENDFILES=yes \
READ=/usr/spool/uucppublic:/u/merlin \
WRITE=/usr/spool/uucppublic:/u/merlin

MACHINE=merlin REQUEST=yes SENDFILES=yes COMMANDS=ALL \
READ=/usr/spool/uucppublic:/u/merlin \
WRITE=/usr/spool/uucppublic:/u/merlin
```

System `merlin` logs in to system `venus` as `umerlin`, which is a unique login for system `merlin`. It can request and send files regardless of who initiated the call. Also, system

merlin can read and write to the **/usr/spool/uucppublic** directory and the **/u/merlin** directory on system venus. It can issue all commands in the default command set on system venus.

Entries on the Remote System

The **Systems** file on merlin contains the following entry for venus, including a phone number and a dialing prefix:

```
venus Any ACU 1200 intown4362 "" in:—in: umerlin word: oaktree
```

System merlin can call system venus at any time, using an ACU device at 1200 baud and logging in as user umerlin with the password oaktree. The telephone number is expanded based on the code intown in the **Dialcodes** file, and the device to be used is determined based on the *Type* and *Class* entries. Accordingly, BNU checks the **Devices** file for a device of type ACU and class 1200.

The **Dialcodes** file on system merlin contains the following dial-code prefix for use with the number in the **Systems** file:

```
intown 9=325
```

Therefore, the expanded telephone number to reach system venus is 9=3254362.

The **Devices** file on system merlin contains the following entry for the connection to venus:

```
ACU  tty1  -  1200  hayes \T
```

The ACU is attached to port tty1, and the dialer is hayes. The telephone number is expanded with information from the **Dialcodes** file. BNU checks the **Dialers** file for an entry for a hayes modem.

The **Dialers** file on system merlin contains the following entry for its modem:

```
hayes  =,-,  ""  \dAT\r\c OK \pATDT\T\r\c CONNECT
```

The **Permissions** file on system merlin contains the following entries specifying system venus's access to merlin:

```
LOGNAME=uvenus VALIDATE=venus SENDFILES=call REQUEST=no \
WRITE=/usr/spool/uucppublic:/u/venus \
READ=/usr/spool/uucppublic:/u/venus

MACHINE=venus READ=/ WRITE=/ COMMANDS=ALL REQUEST=yes \
NOREAD=/usr/lib/uucp:/usr/etc/secure \
NOWRITE=/usr/lib/uucp:/usr/etc/secure
```

In the preceding example, the LOGNAME entry for system venus provides much tighter security than the MACHINE entry for this system. Some transactions cannot take place unless system merlin calls system venus, including:

- Files queued on system merlin cannot be transferred to system venus unless merlin initiated the call;
- When system merlin calls system venus, system venus can read and write to all directories except the **/usr/lib/uucp** directory and the **/usr/etc/secure** directory. When system venus calls system merlin, system venus can write to and read only the public directory and the **/u/venus** directory.

However, no matter who initiated the call, system venus can issue all commands on system merlin.

Files

/usr/lib/uucp directory	Contains all the configuration files for BNU.
/usr/lib/uucp/Devices file	Contains information about available devices.
/usr/lib/uucp/Dialcodes file	Contains dialing code abbreviations.
/usr/lib/uucp/Dialers file	Specifies initial handshaking on a connection.
/usr/lib/uucp/Systems file	Describes accessible remote systems.
/usr/lib/uucp/Permissions file	Describes access permissions for remote systems.
/usr/spool/uucppublic directory	Contains files that have been transferred.

Related Information

Example of a BNU Configuration for a Direct Connection on page 6–40, Example of a BNU Configuration for a TCP/IP Connection on page 6–34.

How to Configure BNU on page 6–23.

Example of a BNU Configuration for a Direct Connection

The following files are set up for a hardwired connection between systems `zeus` and `hera`, where `zeus` is considered the local system and `hera` the remote system. The hardwired device on system `zeus` is `tty5`; on system `hera` it is `tty1`. The speed of the connection is `1200` bps. The login ID for system `zeus` on system `hera` is `uzeus`, and the associated password is `thunder`. The login ID for system `hera` on system `zeus` is `uhera`, and the associated password is `portent`.

Entries in the Local System's Files

The **Systems** file on `zeus` contains the following entry for the remote system `hera`:

```
hera Any hera 1200 - "" \r\d\r\d\r in:—in: uzeus word: thunder
```

This entry specifies that system `hera` can log in to system `zeus` at any time, using a direct connection specified in the **Devices** file. To find the entry in the **Devices** file, BNU uses the third and fourth fields of the **Systems** file. Thus BNU looks for an entry in the **Devices** file with a *Type* of `hera` and a *Class* of `1200`. System `zeus` logs in to system `hera` as user `uzeus` with the password `thunder`.

The **Devices** file on `zeus` contains the following entries in order to connect to the remote system `hera`:

```
Direct  tty5  -  1200  direct
hera    tty5  -  1200  direct
```

These two entries, combined, specify that system `zeus` uses the device `tty5` at `1200` bps to communicate with system `hera`. Note that the *Dialer* in both *Dialer-Token Pairs* fields is `direct`. When connecting to system `hera`, BNU checks the **Dialers** file for a `direct` entry.

The **Dialers** file on system `zeus` contains the following entry for direct connections:

```
direct
```

This specifies that no handshaking is required on the direct connection.

The **Permissions** file on the local system `zeus` contains the following entry specifying the ways in which the remote system `hera` can conduct **uucico** and **uuxqt** transactions with `zeus`:

```
LOGNAME=uhera VALIDATE=hera REQUEST=yes \
SENDFILES=yes MACHINE=hera READ=/ WRITE=/ COMMANDS=ALL
```

This entry specifies that system `hera` logs in as `uhera`. Since the `VALIDATE=hera` option is included, system `hera` cannot log in to system `zeus` with any other login ID, nor can any other remote system use the `uhera` ID. System `hera` can read and write to any directory on system `zeus`, and can send and request files regardless of who initiated the call. System `hera` can also initiate any commands on system `zeus`.

Note: Since the permissions that are granted are the same regardless of which system initiated the connection, the LOGNAME and MACHINE entries have been combined. Separately, they are:

```
LOGNAME=uhera VALIDATE=hera REQUEST=yes \
SENDFILES=yes READ=/ WRITE=/

MACHINE=hera READ=/ WRITE=/ REQUEST=yes COMMANDS=ALL
```

Warning: Providing the permissions in the preceding example is equivalent to giving any user on the remote system a login ID on the local system. Such liberal permissions can jeopardize your security and should usually be given only to well-trusted remote systems at the same site.

Entries in the Remote System's Files

The **Systems** file on system hera contains the following entry for zeus:

```
zeus Any zeus 1200 - "" \r\d\r\d\r in:—in: uhera word: portent
```

This entry specifies that system `hera` can log in to system `zeus` at any time, using a direct connection specified in the **Devices** file. To find the entry in the **Devices** file, BNU uses the third and fourth fields of the **Systems** file. Thus BNU looks for an entry in the **Devices** file with a *Type* of `zeus` and a *Class* of `1200`. System `hera` logs in to system `zeus` as user `uhera` with the password `portent`.

The **Devices** file on system `hera` contains the following entries for communications with `zeus`:

```
Direct  tty1  -  1200  direct
zeus    tty1  -  1200  direct
```

These two entries, combined, specify that system `hera` uses the device `tty1` at `1200` bps to communicate with system `zeus`. Since the *Dialer* is specified as `direct`, BNU checks the **Dialers** file for a `direct` entry.

The **Dialers** file on system `hera` contains the following entry for direct connections:

```
direct
```

This specifies that no handshaking is required on the direct connection.

The **Permissions** file on system `hera` contains the following entries specifying the ways in which `zeus` can conduct **uucico** and **uuxqt** transactions with `hera`:

```
LOGNAME=uzeus VALIDATE=zeus REQUEST=yes \
SENDFILES=yes READ=/ WRITE=/

MACHINE=zeus REQUEST=yes COMMANDS=ALL READ=/ WRITE=/
```

These entries specify that system `zeus` logs in to system `hera` as `uzeus`. Since the `VALIDATE=zeus` option is included, system `zeus` cannot log in to system `hera` with any other login ID, nor can any other remote system use the `uzeus` ID. System `zeus` can read and write to any directory on system `hera`, and can send and request files regardless of who initiated the call. System `zeus` can also initiate any commands on system `hera`.

Warning: Providing the permissions in the preceding example is equivalent to giving any user on the remote system a login ID on the local system. Such liberal permissions can jeopardize your security and should usually be given only to remote systems at the same site.

Files

/usr/lib/uucp directory	Contains all the configuration files for BNU.
/usr/lib/uucp/Devices file	Contains information about available devices.
/usr/lib/uucp/Dialers file	Specifies initial handshaking on a connection.
/usr/lib/uucp/Systems file	Describes accessible remote systems.
/usr/lib/uucp/Permissions file	Describes access permissions for remote systems.
/usr/spool/uucppublic directory	Contains files that have been transferred.

Related Information

Example of a BNU Configuration for a Telephone Connection on page 6–37, Example of a BNU Configuration for a TCP/IP Connection on page 6–34.

How to Configure BNU on page 6–23.

Chapter 7. Generic Data Link Control (GDLC) Environment

The Generic Data Link Control (GDLC) is a generic interface definition that allows both application and kernel users to have a common set of commands to control DLC device managers within the AIX Version 3 system.

Generic Data Link Control (GDLC) Environment Introduction

The GDLC interface specifies requirements for entry point definitions, functions provided, and data structures for all DLC device managers. An example set of DLCs that conform to the GDLC interface are:

- DLCTOKEN (Token-Ring)
- DLCETHER (Standard Ethernet)
- DLC8023 (IEEE 802.3 for Ethernet)
- DLCSDLC (Synchronous Data Link Control)
- DLCQLLC (Qualified Logical Link Control).

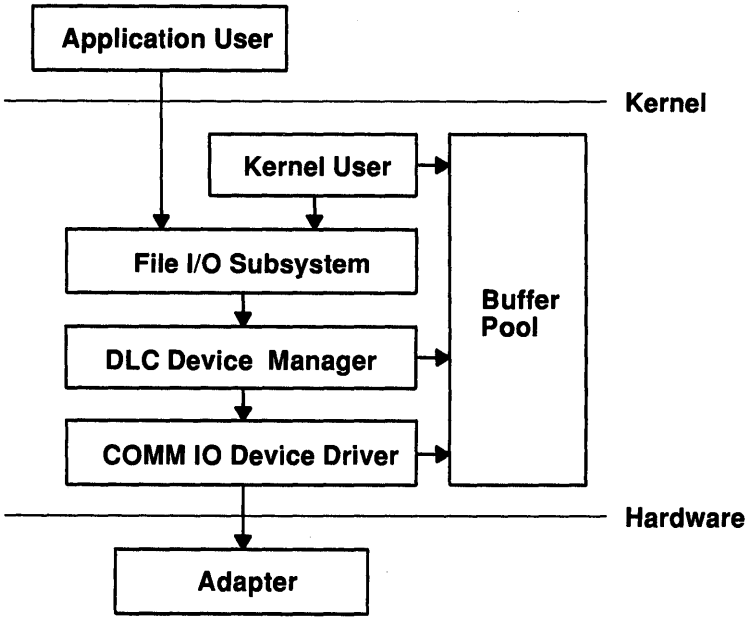
DLC device managers perform higher layer protocols and functions beyond the scope of a kernel device driver, but still reside within the kernel for maximum performance and use a kernel device driver for their I/O requests to the adapter. A DLC user can be located above or within the kernel.

An example of a DLC device manager is Synchronous Data Link Control (SDLC) or IEEE 802.2 Data Link Control. Each DLC device manager operates with a specific device driver or set of device drivers. SDLC, for example, operates with the IBM Multiprotocol device driver for the GL product and its associated adapter.

The basic structure of a DLC environment is shown in the following figure. Users within the kernel have access to the Communications memory buffers (**mbufs**) and call the **dd** entry points by way of the **fp** kernel services. Users above the kernel use the standard interface-to-kernel device drivers, and the file system then calls the **dd** entry points. Data transfers in this case require a move of the data between user and kernel space.

See DLC's environment and its relationship to other components in the following figure.

DLC Device Manager Environment



Application User
Resides above the kernel as an application or access method.

Kernel User Resides within the kernel as a kernel process or device manager.

File I/O Subsystem
Converts the file descriptor and file pointer subroutines to file pointer accesses of the switch table.

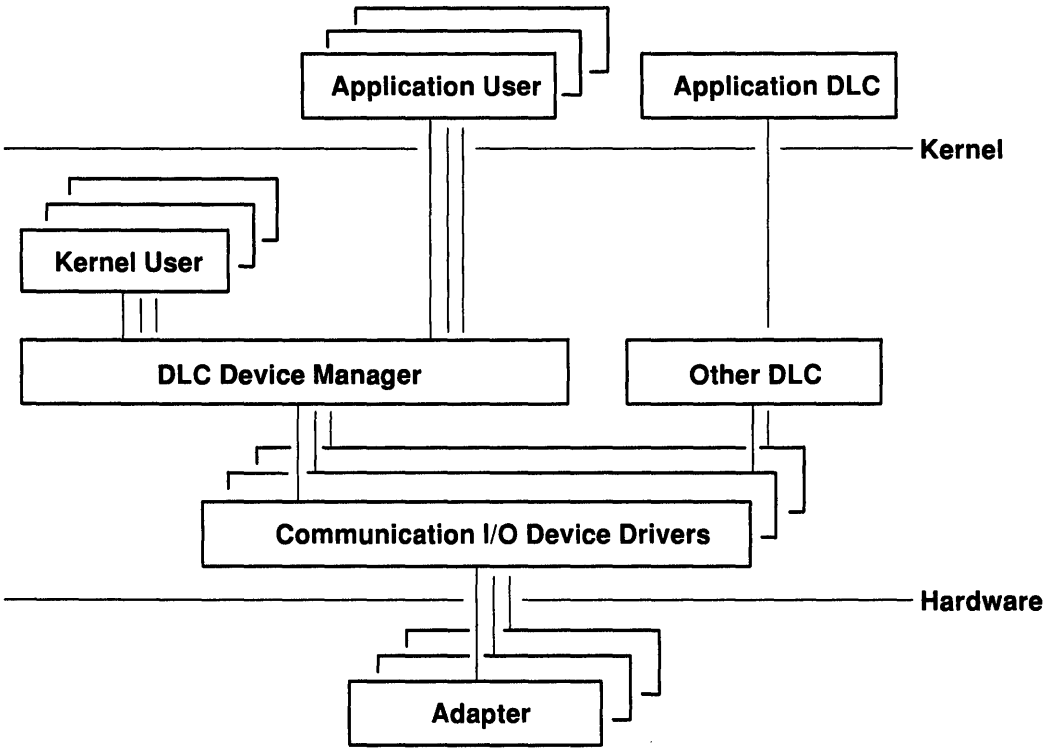
Buffer Pool Provides data buffer services for the communications subsystem.

Comm I/O Device Driver
Controls hardware adapter I/O and DMA registers, and routes receive packets to multiple DLCs.

Adapter Attaches to the communications media.

A device manager written in accordance with GDLC specifications can run on all AIX Version 3 hardware configurations that contain a Communications Device Driver and its target adapter. Each of these device managers can support multiple users above and multiple device drivers and adapters below. In general, users can operate concurrently over a single adapter, or each user can operate over multiple adapters. Some DLC device managers may vary depending on their particular protocol constraints. The following figure illustrates a multiple user configuration.

Example of multiple-user, multiple-adapter configuration



Meeting the GDLC Criteria

There are several criteria that must be met in order for a GDLC interface to be a truly generic interface. This interface must do the following:

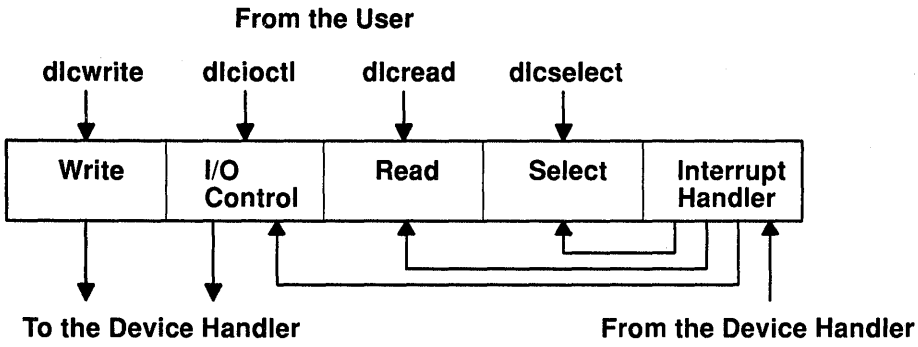
- Be flexible and be accessible to both application and kernel users.
- Have multi-user and multi-adapter capability for protocols that can take advantage of multiple sessions and ports.
- Support both connection-oriented and connectionless types of DLC device managers.
- Provide a means to pass data in a transparent mode for those users that have special requirements beyond the scope of the DLC device manager in use.

Implementing the GDLC Interface

Each DLC device manager is a standard **/dev** entry that operates in the kernel as a multiplexed device manager for a particular protocol. Each **open** subroutine to a DLC device manager for an adapter port that is not already in use by DLC causes a kernel process to be created and an **open** subroutine to be issued to the target adapter's device handler. Additional **open** subroutines can be issued to the same DLC device manager in order to talk to multiple adapter ports of the same protocol. **Open** subroutines that target the same port do not create additional kernel processes, but rather link the new **open** subroutine with the existing process. There is always one kernel process for each port in use.

The internal structure of a DLC device manager has the same basic structure as a kernel device handler, except that the interrupt handler is replaced by a kernel process for handling asynchronous events. The read, write, I/O control, and select blocks function exactly the same, as shown in the following figure.

Standard Kernel Device Manager



Installing Data Link Controls

Each DLC can be installed separately or in a group. Once the DLCs are installed, you must add the DLC device to the system in order to make it useable. You may want to display or list information about an installed DLC to verify that installation and the addition of an installed DLC was successful. Listing current DLC attributes is useful when one must check to see if any changes are necessary. On heavily used systems, fine tuning or changing may be necessary. If receive performance is sluggish and the system error log indicates that the DLC is experiencing ring queue overflows between the DLC and its device handler, the operator can change the DLC's queue depth for incoming data. Finally, you may want to remove an installed DLC from the system when changing networks, such as switching from the SDLC device manager to the Token-Ring device manager. All of these DLC procedures can be accomplished through the System Management Interface Tool (SMIT) install menus or through the command line.

List of the DLC Interface Entry Points

dlcclose Closes a GDLC channel.

dlcconfig Configures the GDLC device manager.

dlcioctl Issues specific commands to the GDLC.

dlcmpx Decodes the device handlers' special file name that was appended to the **open** subroutine.

dlcopen Opens a GDLC channel.

dlcread Reads receive data from the GDLC.

dlcselect Selects asynchronous criteria from the GDLC such as receive data completion and exception conditions.

dlcwrite Writes transmit data to the GDLC.

Understanding DLC Interface ioctl Entry Point Operations

The GDLC interface supports the following **ioctl** subroutine operations:

DLC_ENABLE_SAP
Enables a service access point (SAP).

DLC_DISABLE_SAP
Disables a SAP.

DLC_START_LS
Starts a link station on a particular SAP as caller or listener.

DLC_HALT_LS Halts a link station.

DLC_TRACE Traces a link station's activity for short or long activities.

DLC_CONTACT
Contacts a remote station for a particular local link station.

DLC_TEST Tests the link to a remote for a particular local link station.

DLC_ALTER Alters a link station's configuration parameters.

DLC_QUERY_SAP
Queries statistics of a particular SAP.

DLC_QUERY_LS
Queries statistics of a particular link station.

DLC_ENTER_LBUSY
Enters local busy mode on a particular link station.

DLC_EXIT_LBUSY
Exits local busy mode on a particular link station.

DLC_ENTER_SHOLD
Enters short hold mode on a particular link station.

DLC_EXIT_SHOLD
Exits short hold mode on a particular link station.

DLC_GET_EXCEP
Returns asynchronous exception notifications to the application user.

Note: This **ioctl** subroutine operation is not used by the kernel user since all exception conditions are passed to the kernel user by way of their exception handler.

IOCINFO Returns a structure that describes the GDLC device manager. See **sys/devinfo.h** file format.

Service Access Point (SAP)

A service access point (SAP) identifies a particular user service that will send and receive a specific class of data. This allows different classes of data to be routed separately to their corresponding service handlers. Those DLCs that support multiple concurrent SAPs have SAP addresses known as Destination SAP and Source SAP imbedded in their packet headers. DLCs that can only support a single SAP do not need or use SAP addressing, but still have the concept of enabling the one SAP. In general, there is a SAP enabled for each DLC user on each port.

Most SAP address values are defined by IEEE-standardized network management entities. Some of the common SAP addresses are:

Null SAP (x'00')
Provides some ability to respond to remote nodes even when no SAP has been enabled. This SAP supports only connectionless service and responds only to XID and TEST Link Protocol Data Unit (LPDUs).

SNA Path Control (x'04')
Denotes the default individual SAP address used by SNA nodes.

PC Network NETBIOS (x'F0')
Used for all DLC communication that is driven by NETBIOS emulation.

Discovery SAP (x'FC')
Used by IBM LAN name discovery services.

Global SAP (x'FF')
Identifies all active SAPs.

Link Station (LS)

A link station (LS) identifies an attachment between two nodes for a particular SAP pair. This attachment can operate as a connectionless service (datagram) or connection-oriented service (fully sequenced data transfer with error recovery). In general, there is one LS started for each remote attachment.

Local Busy Mode

Whenever an LS is operating in a connection-oriented mode (contacted) and wishes to stop the remote station's sending of information packets for reasons such as resource outage, notification can be sent to the remote station to cause the local station to enter local busy mode. Once resources are available, the local station will notify the remote that it is no longer busy and that information packets can flow again. Only sequenced information packets are stopped with local busy mode. All other types of data are unaffected.

Short Hold Mode

Use the short hold mode of operation when operating over certain data networks that have the following characteristics:

- Short call setup time
- Tariff structure that specifies a relatively small fee for the call setup compared to the charge for connect time.

With short hold mode an attachment between two stations is maintained only while there is data available for transfer between the two stations. When there is no data to send, the attachment is cleared after a certain time out and established again when there is new data to transfer.

Testing and Tracing a Link

To test an attachment between two stations an LS can be instructed to send a test packet from the local station. This packet is echoed back from the remote station if the attachment is operating correctly.

Some data links may be limited in their support of this function due to protocol constraints. SDLC, for example, only generates the test packet from the host or primary station. Most other protocols, however, allow test packets to be initiated from either station.

To trace a link, line data and special events (such as station activation, termination, and time outs) can be logged in the system's generic trace facility for each link station (LS). This function helps determine the cause of certain communications attachment problems. The GDLC user may select either short or long entries to be traced.

Short entries consist of up to 80 bytes of line data, while long entries allow full packets of data to be traced.

Tracing can be activated when an LS is started, or it may be activated or terminated dynamically anytime after an LS has been started.

Statistics

Both SAP and LS statistics can be queried by a GDLC user. The statistics for a SAP consist of the current SAP state and information about the device handler. LS statistics consist of the current station states and various Reliability/Availability/Serviceability counters that have monitored the activity of the station since it was started.

Using Special Kernel Services

GDLC provides special services for a kernel user, with the understanding that a trusted environment must exist within the kernel. Instead of the DLC device manager copying asynchronous event data into user space, the kernel user must specify function pointers to special routines called function handlers. The function handlers will be called by the DLC at the time of execution. This allows maximum performance between the kernel user and the DLC layers. Each kernel user is required to keep the number of function handlers down to a minimum path length and to use the communications memory buffer (**mbuf**) scheme.

A function handler must never call another DLC entry directly, since the call would be made under lock, causing a fatal sleep. The only exception to this general rule is that a kernel user is allowed to call the **dlcwritex** entry during its service of any of the four receive data functions. Calling the **dlcwritex** entry allows immediate responses to be generated without an intermediate task switch. Special logic is required within the DLC device manager to check the process identification of the user calling a write. If it is a DLC process and the internal queueing capability of the DLC has been exceeded, the write is sent back with a bad

return code (EAGAIN return value) instead of putting the calling process (DLC) to sleep. It is then up to the calling user subroutine to return a special notification to DLC from its receive data function to ensure a retry of the receive buffer at a later time.

The user provided function handlers are:

Datagram Data Received Routine
Called any time a datagram packet is received for the kernel user.

Exception Condition Routine
Called any time an asynchronous event occurs that must notify the kernel user, such as SAP Closed or Station Contacted.

I-Frame Data Received Routine
Called each time a normal sequenced data packet is received for the kernel user.

Network Data Received Routine
Called any time network-specific data is received for the kernel user.

XID Data Received Routine
Called any time an exchange identification (XID) packet is received for the kernel user.

The **dlcread** and **dlcselect** entry points for DLC are not called by the kernel user because the asynchronous functional entries are called by the DLC device manager directly. Generally, any queuing of these events must occur in the user's function handler. If, however, the kernel user cannot handle a particular receive packet, the DLC device manager may hold the last receive buffer and enter one of two special user busy modes.

User Terminated Busy Mode (I-frame only)
If the kernel user cannot handle a received I-frame (due to problems such as queue blockage), a –1 return code is given back and DLC holds the buffer pointer and enters local busy mode to stop the remote station's I-frame transmissions. The kernel user must call the Exit Local Busy function to reset local busy mode and start the reception of I-frames again. Only normal sequenced I-frames can be stopped. XID, datagram, and network data are not affected by local busy mode.

Timer Terminated Busy Mode (all frame types)
If the kernel user cannot handle a particular receive packet and wants DLC to hold the receive buffer for a short period and then re-call the user's receive function, a –2 return code is given back to DLC. If the receive packet is a sequenced I-frame, the station enters local busy mode for that period. In all cases, a timer is started, and once the timer expires, the receive data functional entry is called again.

Understanding DLC Problem Determination

Each of the generic data link controls provide problem determination data that can be used to isolate network problems. Three types of diagnostic information are provided:

- Status
- Error Log
- Link Trace.

Understanding DLC Status Information

Status can be obtained for a service access point (SAP) or a link station using DLC_QUERY_SAP and DLC_QUERY_LS **ioctl** subroutines to the specific DLC kernel device manager in use.

Individual device driver statistics can be obtained with the DLC_QUERY_SAP **ioctl** subroutine from various devices such as:

- Token-Ring
- Ethernet
- Multiprotocol
- X.25.

Link station statistics can be obtained with the DLC_QUERY_LS **ioctl** subroutine from various data link controls. These statistics include data link protocol counters. Each counter is reset by the DLC during the DLC_START_LS **ioctl** subroutine and generally runs continuously until the link station is terminated and its storage is freed. If a counter reaches the maximum count, the count is frozen and no wrap around occurs.

The suggested counters to be provided by a DLC device manager are shown below. Some DLCs may wish to modify this set of counters based on the specific protocols being supported. For example, the number of rejects or receive-not-ready packets received might be meaningful.

Test Commands Sent

Contains a binary count of the **test** commands sent to the remote station by GDLC, in response to **test** commands issued by the user.

Test Command Failures

Contains a binary count of the **test** commands that did not complete properly due to problems such as the following:

- Invalid response
- Bad data compare
- Inactivity.

Test Commands Received

Contains a binary count of valid **test** commands received, regardless of whether the response is completed properly.

Sequenced Data Packets Transmitted

Contains a binary count of the total number of normal sequenced data packets that were transmitted to the remote link station.

Sequenced Data Packets Transmitted

Contains a binary count of the total number of normal sequenced data packets that were retransmitted to the remote link station.

Maximum Contiguous Retransmissions

Contains a binary count of the maximum number of times a single data packet has been retransmitted to the remote link station prior to acknowledgment. This counter is reset each time a valid acknowledgment is received.

Sequenced Data Packets Received
Contains a binary count of the total number of normal sequenced data packets that have been correctly received.

Invalid Packets Received
Contains a binary count of the number of invalid commands or responses received, including invalid control bytes, invalid I-fields, and overflowed I-fields.

Adapter Detected Receive Errors
Contains a binary count of the number of receive errors reported back from the device driver.

Adapter Detected Transmit Errors
Contains a binary count the number of transmit errors reported back from the device driver.

Receive Inactivity Timeouts
Contains a binary count of the number of receive time outs that have occurred.

Command Polls Sent
Contains a binary count of the number of command packets sent, that requested a response from the remote link station.

Command Repolls Sent
Contains a binary count of the total number of command packets that were retransmitted to the remote link station due to lack of response.

Command Contiguous Repolls
Contains a binary count of the number of times a single command packet was retransmitted to the remote link station due to lack of response. This counter is reset each time a valid response is received.

Understanding the DLC Error Log

Each DLC provides entries to the system error log whenever errors are encountered. To call the kernel error collector, use the **errsave** kernel service.

GDLC supports the GL product Network Management Alert Management architecture for reporting error conditions. The error conditions are reported by the GL product system error log using the error log daemon (**errdaemon**). Each error is defined with the following entries:

Error Type Indicates the severity of the error. The three levels of severity are as follows:

Temporary Indicates errors that do not force closure of the link station or SAP connections but are logged for network analysis.

Permanent Indicates errors that result in the closure of the individual link station, SAP, or the entire physical port due to their catastrophic nature.

Performance Indicates errors such as queue overruns that are causing performance degradation due to retransmissions and other factors.

Error Description
Describes the failure.

Probable Cause
Describes what likely caused the failure.

Recommended Actions
Describes how to correct the problem. This is divided into operator (User), installation/set-up (Install), and resource (Failure) actions.

Detailed Data Provides additional data obtained at the time of the error.

The user can obtain formatted error log data by issuing the **errpt** command. When used with the **-R** *DLCType* flag, the **errpt** command produces a detailed report of all the error log entries for the resource type indicated by the *DLCType* variable previously collected in the **/etc/rasconf** default file. Valid values for the *DLCType* variable include:

dlcether Standard Ethernet datalink

dlc802.3 IEEE 802.3 Ethernet datalink

dlctoken Token-Ring datalink

dlcsdlc SDLC datalink.

Alerts Some error conditions must generate specific error log formats so that Alert Vectors can be sent to the local network manager and possibly to a remote SNA host.

The format of each required Alert Vector can be found in Appendix A of IBM publication *SNA Format and Protocol Reference Manual: Management Services (SC30-3346).*

Understanding the DLC Link Trace Facility

GDLC provides optional entries to a generic system trace channel as required by the GL product system Reliability/Availability/Serviceability. GDLC is defaulted with trace disabled in order to provide maximum performance and reduce the number of system resources utilized. For more information, see Understanding the DLC Local Area Network Monitor Trace.

Trace Channels

AIXv3 supports up to seven generic trace channels in operation simultaneously. A channel must be allocated by the user prior to activation of a link trace, whether it is being started in the DLC_START_LS **ioctl** operation or in the DLC_TRACE **ioctl** operation. This is accomplished with the **trcstart** and **trcon** subroutines.

Trace activity in the link station must be stopped by either halting the link station or by issuing an **ioctl**(DLC_TRACE, flags=0) operation to that station. See DLC_TRACE ioctl operation for DLC. Once the link station has stopped tracing, the channel can be disabled using the **trcoff** subroutine and returned to the system using the **trcstop** subroutine.

Trace Reports

The user can obtain formatted trace log data by issuing the **trcrpt** command with the appropriate file name, such as:

```
trcrpt /tmp/link1.log
```

This example produces a detailed report of all the link trace entries in the **/tmp/link1.log** file, *if* a prior **trcstart** subroutine specified the **/tmp/link1.log** file as the (**-o**) name for the trace log.

Trace Entries

The subroutine call generated by GDLC to the kernel Generic Trace for each entry is:

#include <sys/trchkid.h>

void trcgenkt (*chan*, *hk_word*, *data_word*, *len*, *buf*)
unsigned int *chan*, *hk_word*, *data_word*, *len*;
char **buf*;

where:

chan Specifies the channel number for the trace session. This number is obtained from the **trcstart** subroutine.

hk_word Contains the trace hook identifier as defined in the **/usr/include/sys/trchkid.h** header file.

Five types of link trace entries are registered using hook ID:

HKWD_SYSX_DLC_START Start Link Station Completions

HKWD_SYSX_DLC_TIMER Timeout Completions

HKWD_SYSX_DLC_XMIT Transmit Completions

HKWD_SYSX_DLC_RECV Receive Completions

HKWD_SYSX_DLC_HALT Halt Link Station Completions.

data_word Specifies trace data format field which varies depending on the hook ID. Each of these definitions can be found in the **/usr/include/sys/gdlextcb.h** header file.

The first half-word always contains the data link protocol field.

DLC_DL_SDLC SDLC

DLC_DL_HDLC HDLC

DLC_DL_BSC BISYNC

DLC_DL_ASC ASYNC

DLC_DL_PCNET PC Network

DLC_DL_ETHER Standard Ethernet

DLC_DL_802_3 IEEE 802.3

DLC_DL_TOKEN Token-Ring

DLC_DL_QLLC X.25 Qualified DLC.

The second half-word is as follows:

- On start or halt link station completions, the second half-word contains the physical link protocol in use:

  DLC_PL_EIA232 EIA-232D Telecommunications

  DLC_PL_EIA366 EIA-366 Auto Dial

  DLC_PL_X21 CCITT X.21 Data Network

  DLC_PL_PCNET PC Network Broadband

  DLC_PL_ETHER Standard Baseband Ethernet

  DLC_PL_SMART Smart Modem Auto Dial

  DLC_PL_802_3 IEEE 802.3 Baseband Ethernet

  DLC_PL_TBUS IEEE 802.4 Token Bus

  DLC_PL_TRING IEEE 802.5 Token-Ring

  DLC_PL_X25 X.25 Packet Network

  DLC_PL_EIA422 EIA-422 Telecommunications

  DLC_PL_V35 CCITT V.35 Telecommunications

  DLC_PL_V25BIS CCITT V.25 bis Autodial for Telecommunications.

- On time out completions, the second half-word contains the type of time out occurrence:

  DLC_TO_SLOW_POLL
  Slow Station Poll

  DLC_TO_IDLE_POLL
  Idle Station Poll

  DLC_TO_ABORT Link Station Aborted

  DLC_TO_INACT Link Station Receive Inactivity

  DLC_TO_FAILSAFE Command Failsafe

  DLC_TO_REPOLL_T1
  Command Repoll

  DLC_TO_ACK_T2 I-frame Acknowledgment.

- On transmit completions, the second half-word is set to the data link control bytes being sent. Some transmit packets only have a single control byte, so the second control byte is not displayed in those cases.

- On receive completions, the second half-word is set to the data link control bytes that were received. Some receive packets only have a single control byte, so the second control byte is not displayed in those cases.

len Specifies the length in bytes of the entry specific data specified by the *buf* parameter.

buf Specifies the pointer to the entry specific data that consists of:

Start Link Station Completions
Link Station Diagnostic Tag and the remote station's name and address.

Timeout Completions
No specific data is recorded.

Transmit Completions
Either the first 80 bytes or all the transmitted data, depending on the short/long trace option.

Receive Completions
Either the first 80 bytes or all the received data, depending on the short/long trace option.

Halt Link Station Completions
Link Station Diagnostic Tag, the remote station's name and address, and the result code.

Data Link Control (DLC) Reference Information

DLC Entry Points

dlcclose Entry Point of the GDLC Device Manager.

dlcconfig Entry Point of the GDLC Device Manager.

dlcioctl Entry Point of the GDLC Device Manager.

dlcmpx Entry Point of the GDLC Device Manager.

dlcopen Entry Point of the GDLC Device Manager.

dlcread Entry Point of the GDLC Device Manager.

dlcselect Entry Point of the GDLC Device Manager.

dlcwrite Entry Point of the GDLC Device Manager.

Kernel Services for DLC

fp_close Kernel Service for Data Link Control (DLC) Devices.

fp_ioctl Kernel Service for Data Link Control (DLC) Devices.

fp_open Kernel Service for Data Link Control (DLC) Devices.

fp_write Kernel Service for Data Link Control (DLC) Devices.

DLC Kernel Routines for DLC

DLC Datagram Data Received Routine.

DLC Exception Condition Routine.

DLC I-Frame Data Received Routine.

DLC Network Data Received Routine.

DLC XID Data Received Routine.

Subroutines Available for DLC

DLC Extended Parameters for Subroutines

DLC Extended Parameters for **open** Subroutine.

DLC Extended Parameters for **read** Subroutine.

DLC Extended Parameters for **write** Subroutine.

Application Subroutines

close Subroutine Interface for Data Link Control (DLC) Manager.

ioctl Subroutine Interface for Data Link Control (DLC) Manager.

open Subroutine Interface for Data Link Control (DLC) Manager.

readx Subroutine Interface for Data Link Control (DLC) Manager.

select Subroutine Interface for Data Link Control (DLC) Manager.

writex Subroutine Interface for Data Link Control (DLC) Manager.

DLC Operations

Ioctl Operations (op) for DLC.

Parameter Blocks by Operation for DLC.

Related Information

The **errpt** command.

The **errsave** kernel service, **trcgenkt** kernel service.

The **trcoff** subroutine, **trcon** subroutine, **trcstart** subroutine, **trcstop** subroutine.

The **icotl** subroutine for DLC, **open** subroutine for DLC.

How to List Current DLC Information

Prerequisite Task

A DLC must be installed and added before you can list or show its current attributes.

Procedure

The user can query a DLC using the System Management Interface Tool (SMIT) menus, the SMIT fast path, or the command line.

- To list current DLC information through the SMIT fast path, enter the SMIT fast path command followed by the device manager of your choice:

  ```
  smit cmddlc_ether
  ```

  Note: The SMIT fast path to list or show a device manager includes both Standard Ethernet and IEEE 802.3 Ethernet device managers.

  OR

  ```
  smit cmddlc_sdlc
  ```

  OR

  ```
  smit cmddlc_token
  ```

  OR

  ```
  smit cmddlc_qllc
  ```

  Entering the SMIT fast path will allow you to add, change, show, or remove a DLC device manager. Select the change/show option.

- A more versatile way to display information about DLCs is to issue the **lsdev** or **lsattr** commands. These commands can be invoked as follows:

  – To list all the predefined DLC types installed or just those DLC types that connect to a specified adapter subclass, use the **lsdev** command in the following format:

    ```
    lsdev -P -c dlc
    ```

    The **-P** flag causes all devices designated by the **-c** *Class* variable in the predefined data base to be listed by type name. Output is given in the following format:

    ```
    dlc  IEEE_ethernet dlc  IEEE Ethernet (802.3) Data Link Control
    dlc  ethernet      dlc  Standard Ethernet Data Link Control
    dlc  X25_qllc      dlc  X.25 QLLC Data Link Control
    dlc  sdlc          dlc  SDLC Data Link Control
    dlc  tokenring     dlc  Token-Ring Data Link Control
    ```

  – To list all the customized DLCs already in the system or specific DLCs identified by logical name (also called *DeviceName*), use the **lsdev** command in the following format:

    ```
    lsdev -C -c dlc
    ```

The **–C** flag causes all devices designated by the **–c** *Class* variable in the customized data base to be listed by logical name (also called *DeviceName*). Output is given in the following format:

```
dlc8023 Available           IEEE Ethernet (802.3) Data Link Cont
rol
dlcether      Available     Standard Ethernet Data Link Control
dlcqllc Available           X.25 QLLC Data Link Control
dlcsdlc Available           SDLC Data Link Control
dlctoken      Available     Token–Ring Data Link Control
```

– To list all predefined or customized attribute information related to a DLC or DLCs, use the **lsattr** command in the following format:

```
lsattr –D –l dlctoken
```

The **–D** flag causes all the default attributes of the **–l** *Name* variable to be listed. The **–E** flag lists the currently effective attribute values and the **–R** flag lists the possible range of values. Output is given in the following format:

```
queue_depth     32    Depth of the receive queue
```

Related Information

The **lsattr** command, **lsdev** command.

How to Install AIX for RISC System/600 Licensed Program Optional Program Products in *General Concepts and Procedures.*, How to Change DLC Attributes on page 7–18, How to Remove a DLC on page 7–20.

SMIT Overview in *General Concepts and Procedures.*

How to Change DLC Attributes

Prerequisite Tasks or Conditions

1. A DLC must be installed and added before you can change its attributes.
2. An attribute change is only successful if there are no active opens against the target DLC. Prior to issuing the change action, the user may have to halt usage of the DLC by such services as SNA, OSI, or NETBIOS.
3. Because it directly affects system resources, change the receive queue depth or size only if the receive performance is sluggish or the system error log indicates that the DLC is experiencing receive queue overflows between the DLC and its device handler.

Procedure

Heavily used systems may require fine tuning of a DLC. This is accomplished using the System Management Interface Tool (SMIT) menus, the SMIT fast path, or the command line.

- To use the SMIT fast path in order to change DLC information, enter the SMIT fast path command followed by the device manager of your choice:

  ```
  smit cmddlc_ether
  ```

  Note: The SMIT fast path to change a device manager includes both Standard Ethernet and IEEE 802.3 Ethernet device managers.

  OR

  ```
  smit cmddlc_sdlc
  ```

  OR

  ```
  smit cmddlc_token
  ```

  OR

  ```
  smit cmddlc_qllc
  ```

  Entering the SMIT fast path will allow you to add, change, show, or remove a DLC device manager. Select the change/show option.

  The change/show option displays the attributes of the *DeviceName* DLC and allows the receive queue depth attribute to be increased from its default value of 32 queue entries. This value should be increased only in small increments since it directly affects system resources.

- The DLC queue depth can also be changed by issuing the **chdev** command. This command causes the target DLC's receive queue to be altered in the same way the SMIT Change/Show *DeviceName* changed a DLC.

  For example, to change the token-ring DLC's receive queue depth, enter the following:

  ```
  chdev -l dlctoken -a queue_depth=50
  ```

  In the preceding example, the value of the **-l** *Name* variable is set to `dlctoken`, and the new receive queue depth is set to 50 queue entries. If the **chdev** command is successful, the system will respond with the following message:

  ```
  dlctoken changed
  ```

Related Information

The **chdev** command.

SMIT Overview in *General Concepts and Procedures*.

How to Install AIX for RISC System/600 Licensed Program Optional Program Products in *General Concepts and Procedures*, How to List Current DLC Information on page 7–16, How to Remove a DLC on page 7–20.

How to Remove a DLC

Prerequisite Tasks or Conditions

1. A DLC must be installed (refer to the procedure found in *General Concepts and Procedures*) and added before you can remove it from the system.
2. Removing a DLC is only successful if there are no active opens against the target DLC. Prior to issuing the remove action, the user may have to halt usage of the DLC by such services as SNA, OSI, or NETBIOS.

Procedure

Remove the DLCs from the system using the System Management Interface Tool (SMIT) menus, the SMIT fast path, or the command line.

- To remove a DLC through the SMIT fast path, enter the SMIT fast path command followed by the device manager of your choice:

  ```
  smit cmddlc_ether
  ```

  Note: The SMIT fast path to remove a device manager includes both Standard Ethernet and IEEE 802.3 Ethernet device managers.

  OR

  ```
  smit cmddlc_sdlc
  ```

  OR

  ```
  smit cmddlc_token
  ```

  OR

  ```
  smit cmddlc_qllc
  ```

  Entering the SMIT fast path allows you to add, change, show, or remove a DLC device manager. Select the remove option.

  The remove option undefines and unconfigures the *DeviceName* DLC from the system and removes a **/dev/***DeviceName* special file.

- DLCs are also deleted from the system by issuing a **rmdev** command. This command unconfigures and (optionally) undefines the target DLC from the system in the same way the SMIT Delete *DeviceName* removes a DLC.

  For example, to remove the token-ring DLC, enter the following:

  ```
  rmdev -d -l dlctoken
  ```

  In this example, the **-d** flag undefines and unconfigures the DLC and sets the value of the **-l** *Name* variable to `dlctoken`. If the **rmdev** command succeeds, the system responds with the following message:

  ```
  dlctoken deleted
  ```

Related Information

The **rmdev** command.

How to Change DLC Attributes on page 7–18, How to Install AIX for RISC System/600 Licensed Program Optional Program Products in *General Concepts and Procedures*, How to List Current DLC Information on page 7–16.

SMIT Overview in *General Concepts and Procedures*.

Chapter 8. Mail Facilities

The Mail system is a general purpose, internetwork mail-routing facility. This system is not tied to any one transport protocol. It relays messages from one user to another across system and network domain boundaries. While processing the messages, the mail system can do a limited amount of message-header editing to put the message into a format that is appropriate for the receiving domain.

Mail Overview

You can create and send mail and files to users who are on your local system, or to users on remote systems which are connected to your computer. Mail usually takes the form of a letter, but can also be files. You can also send system messages, secret mail, and vacation messages. System messages are usually short and can be used to inform users that the system has been updated. They are similar to broadcast messages, but they are sent to users on your local system only. Secret mail is used to send classified information to someone. After you enter the secret mail message it is encrypted and the receiver must enter a password in order to read it. Vacation messages are used to inform users that you are on vacation. When mail is sent to you a message is sent to the sender of the mail that you are on vacation. Any mail that you receive while on vacation can also be forwarded to someone else.

Sending mail on the computer is similar to the delivery of letters through a national postal system. Each system has a unique address associated with it. You can create nickname, alias, or distribution files to be used for sending mail and mail messages.

When you get mail from someone, you have choices of what to do with it. You can leave it in the system mailbox, read and delete it, forward it to someone else, add comments to it, store it in your personal mailbox (mbox in the **mail** program), or store it in a mail folder that you have created.

There are several programs you can use to send and receive mail. Which program you use depends on what is installed and configured for you. The following programs can be used to send and receive mail:

- The Mail program
- The MH program
- The Bellmail program
- The xtalk program.

Creating and Sending Mail

You can use the **mail** command in one of two ways to send information. You can use the **mail** command's built–in editor to both compose and send a message. You can also use the **mail** command to send any text file to another user. The file may be a letter you have written using your favorite editor, a source file for a program you have written, or any other file in text format.

You can send messages and files to another user on your local system, on another system connected to your system in a network, or on another system connected to another network that has a connection to your network.

You can use the **mail** program to send information to another user. Mail can be one line or many pages. Mail that you create can go to one or more people who are on your local system or a remote system which is connected to your system. The other user does not need to be logged onto the system when you send the information. Previously created files can also be sent using mail.

When you create mail with the **mail** command editor, you are provided with fields in which you enter information such as: date, subject, and cc. These fields are similar to the body of a standard business letter.

The **mail** command provides a line-oriented editor for composing messages. This editor allows you to enter each line of the message and then press the Enter key to get a new line to enter more text. You cannot change the text after you press the Enter key. However, before you press the Enter key, you can change information on that one line by using the Backspace key and Erase key sequence to erase the information and then type in the correct information. Although you cannot change information on a line once you have pressed the Enter key and gone on to the next line, you can change the contents of your message before sending it by using the **visual** or **edit** subcommands to edit the message.

Mail and mail messages are sent to a user's address. The address contains the person's user ID, system name, and is similar to a street address. It tells the system where mail and mail messages should be delivered. If you are sending mail to someone on your local system, you do not have to enter an address. For example, to send mail to user jan on your local system, you would enter:

mail jan

Receiving and Handling Mail

When other users send mail to you, it is stored in a directory that is specifically for mail. This system directory contains a file for every user on the local system. This directory holds your mail until you do something with it. A message notifies you when you have mail. When mail is received and the **mail** command is issued, a list of all unopened mail is displayed.

When mail arrives for you from another user, the **mail** system puts the mail in your system mailbox (**/usr/spool/mail/***UserID*). The command shell will notify you that mail has arrived before displaying its next prompt, provided that the **MAIL** environment variable it set and the interval specified by **MAILCHECK** has elapsed since the shell last checked for mail. The notification message is the value of the **MAILMSG** environment variable. Depending on which shell you are using (Bourne, K, or C shell), the notification message might be:

```
[YOU HAVE NEW MAIL]

Mail  [5.2 UCB]  [IBM AIX 3.1]  Type ? for help.
"/usr/mail/lance": 2 messages 2 new
>N  1 karen      Thu Sep 17 14:36  13/359 "Dept Meeting"
 N  2 karen      Thu Sep 17 16:28  13/416 "Dept Meeting Delayed"
&
```

The display fields in the above example are:

Pointer	The > in this field for a particular message indicates that the message is the current message in the mailbox. If no other message number is specified, the current message is the default message.

Status	The one-letter indicator of the status of the message: `M` Indicates that the message will be stored in your personal mailbox. `N` Indicates that the message is a new message. `P` Indicates that the message will be held (preserved) in your system mailbox. `R` Indicates that you have read the message. `U` Indicates that the message is an unread message. The message has been listed in the mailbox before, but you have not looked at the contents of the message. `*` Indicates that you have saved or written the message to a file or folder. No indicator indicates that the message has been read but has not been deleted or saved.
Message number	The number that mailbox commands use to refer to the message.
Address	The address of the person that sent the message.
Date	The date the message was received, including day of the week, month, date, and time.
Size	Size of the message in number of lines and number of characters, including heading information.
Subject	The contents of the subject field of the message (if the message has one).

From the mailbox prompt `&`, you can enter subcommands to look at, reply to, save, discard, or otherwise manage the contents of the mailbox. Many mailbox subcommands allow you to specify groups of messages upon which to perform the subcommand. Subcommands that allow groups of messages use the parameter *MessageList* in the command format. For example, the format of the **from** subcommand (display information about messages) is displayed as:

```
from [MessageList]
```

In this format, *MessageList* can be one of the following:

- One or more message numbers separated by spaces

  ```
  & f 1 2 4 7
  ```

- A range of message numbers indicated by the first and last numbers in the range separated by a - (hyphen):

  ```
  & f 2–5
  ```

  is the same as:

  ```
  & f 2 3 4 5
  ```

- One or more addresses separated by spaces to apply the subcommand to messages received from those addresses:

  ```
  & f karen lance@zeus
  ```

  The characters entered for an address do not have to exactly match the address. They must only be contained in the address field of the messages in either uppercase or lowercase. Therefore, the request for address `karen` matches all of the following addresses (and many others):

  - karen
  - Karen
  - Karen@zeus
  - skaren

- A string, preceded by a slash, to match against the subject field of the messages:

  ```
  & f/meet
  ```

  applies the subcommand to all messages whose subject field contains the letters `meet` in uppercase or lowercase. The characters entered for a match pattern do not have to exactly match the subject field. They must only be contained in the subject field of the messages in either uppercase or lowercase. Therefore, the request for subject `meet` matches all of the following subjects (and many others):

  - Meeting on Thursday
  - Come to meeting tomorrow
  - MEET ME IN ST. LOUIS

The special character `.` (period) addresses all messages, `*` (asterisk) addresses the first message, and `$` (dollar sign) addresses the last message. All commands that take a message list will default to the current message number if no list is specified. Note that this means that all MessageList parameters should be enclosed in "[ ]", since the list is always optional.

Using the **mail** subcommands on the command line, you can do the following:

- Read it in any order (the default is the newest first)
- Delete it
- Save it to a file
- Forward it to someone else
- Display a list of things you can do with it
- Ignore it
- Append to it
- Answer or reply
- Leave it in the system mailbox
- Store it in your personal mailbox (mbox)
- Store it in a mail file that you have created.

Using Mailboxes

The system mailbox is similar in concept to the postal mailbox into which the post office delivers letters addressed to a person. In the IBM AIX Base Operating System mail program, the system mailbox is a file assigned to a particular user. The file is created when mail arrives for a user ID, and it is deleted when all messages have been removed. A separate system mailbox exists for each user ID defined in the **/etc/passwd** file. The Mail system keeps all system mailboxes in the directory **/usr/spool/mail**. Each system mailbox is named by the user ID associated with it. For example, if your user ID is karen, then your system mailbox is **/usr/spool/mail/karen**.

Use the **mail** command to read and remove messages from your system mailbox. Do not use the system mailbox to store messages; store messages in your personal mailbox and in folders.

Your personal mailbox is similar in concept to an in-basket in an office. You put mail in the in-basket after you have received it, but before you have filed it. The personal mailbox is a working storage place for mail that still requires action.

In the mail system, the personal mailbox is a file assigned to a particular user. The mail system creates the file with the name **$HOME/mbox** (where **$HOME** is the user's login directory) when the user receives mail from the system mailbox. The **mbox** file only exists when it contains a message.

When you use the **mail** program to view mail in your system mailbox, The **mail** program automatically puts all messages that you have read, but did not delete, into your personal mailbox. The messages remain in your personal mailbox until you move or delete them.

Personalizing the Mail Program

The contents of the **/usr/lib/Mail.rc** file defines the configuration of the **mail** program. You can alter the system configuration for your mail program by creating a **$HOME/.mailrc** file. Subcommands in this file override similar subcommands in the **/usr/lib/Mail.rc** file when you run the **mail** program, and these options are valid each time you use the **mail** program. Some of the characteristics of a mail session that you can change include:

- Whether **mail** prompts for the subject of a message
- Whether **mail** prompts for users to get a copy of a message
- If any aliases or distribution lists are defined
- How many lines are displayed when reading messages
- What information is listed in messages
- Where to store messages
- Whether a log file is set up to record outgoing messages
- Whether different editors can be used for entering messages
- How to exit the **mail** editor
- How **mail** stores messages.

Changing the Information Displayed at the Top of a Message

You can use the **ignore** subcommand to suppress message header fields that are normally displayed when you read a message using the **type** subcommand or the **print** subcommand. Several header fields, such as Message–ID and Received, are not generally of interest, and you may therefore choose to ignore them. Add the fields you want to ignore to your **$HOME/.mailrc** file.

The format for the **ignore** subcommand is:

```
ignore [FieldList]
```

There are four **mail** subcommands that are most commonly used to alter the characteristics of the **mail** session. These are the **set** subcommand, **unset** subcommand, **alias** subcommand, and the **ignore** subcommand. The **set** and **unset** subcommands enable and disable mail options, the **alias** subcommand shortens how you address mail, and the **ignore** subcommand suppresses message header fields.

Enabling and Disabling Options

Use the **set** subcommand to enable options and the **unset** subcommand to disable options. Options can be either binary or valued. Binary options are either **set** or **unset**, while valued options can be set to a specific value.

The format for the **set** subcommand to enable options is:

```
set [option_list | Option=Value]
```

The format for the **unset** subcommand to disable options is:

```
unset [OptionList]
```

The following is a list of the options that can be set or unset:

Option	Function
append	Causes messages saved in your **$HOME/mbox** file to be added to the end rather than to the beginning of the mailbox.
ask	Causes you to be prompted for the subject of each message you send. If you respond by pressing the Enter key, no subject field is set.
askcc	Causes you to be prompted for the addresses of people to receive copies of the message. If you respond by pressing the Enter key, no additional copies will be sent.
autoprint	Causes the **delete** subcommand to delete the current message and display the next message.
debug	Causes the **mail** program to display debugging information. Messages are not sent while in debug mode. Same as specifying **–d** flag on the command line.
dot	Causes the **mail** program to interpret a period entered on a line by itself as the end of a message you are sending. After you exit the mail editor, the message is sent, and you return to the mailbox prompt (&).
hold	Holds messages that you have read but have not deleted or saved in the system mailbox (**/usr/spool/mail/***UserID*) instead of in your personal mailbox (**$HOME/mbox**). This option has no effect on deleted messages.

ignore Causes interrupt signals from your terminal to be ignored and echoed as @'s.

ignoreeof Related to the **dot** subcommand. Makes the **mail** program refuse to accept Ctrl–D as the end of a message or as the **quit** subcommand.

metoo Usually, when an alias containing the sender is expanded, the sender is removed from the expansion. Setting this option causes the sender to be included in the alias expansion and thus receive copies of messages. With this entry in your **$HOME/.mailrc** file, when you send a message using an alias name that includes you, and a copy of the message will be put in your mailbox.

nosave Normally, when a message is terminated with two interrupt sequences (Ctrl–C), the **mail** program copies the partial letter to the **dead.letter** file in your home directory. Setting the **nosave** option prevents this.

quiet Suppresses the printing of the program banner when the **mail** program starts. The banner is the line that shows the name of the **mail** program.

Replyall Reverses the sense of the **reply** subcommand and the **Reply** subcommand.

verbose Same as specifying the **–v** flag on the command line. When the **mail** program runs in verbose mode, the actual delivery of messages is displayed on the user's terminal.

The following is a list of the options that can to be assigned a value:

Option	Function
crt=*Number*	Enables you to use the **paging** program to browse through long messages. The value for *Number* determines how many lines a message must be before the **paging** program is started. The **paging** program is invoked whenever you read messages with more than this many lines.

EDITOR=*PathName*

Defines the editor that you start with the **~e** and **edit** subcommands. The value of *PathName* must be the full path name to the editor program that you want to use. The default is **/usr/bin/e**.

escape=*Character*

If defined, the first character of this option is the character to use in place of the ~ character to denote mail editor subcommands.

folder=*PathName*

Defines the name of the directory to use for storing folders of messages. If this name begins with a / (slash), the **mail** program considers it to be an absolute path name; otherwise, the folder directory is found relative to your home directory.

PAGER=*PathName*

Path name of the paging program to use for the **more** subcommand or when the **crt** variable is set. If you do not specify a value for the **PAGER** option, the system uses **/bin/pg**.

record=*PathName*

If defined, gives the path name of the file (relative to **$HOME**) used to record all outgoing mail. A copy of all the messages you send out will be saved in this file. Review this file periodically and delete all unnecessary messages. The **mail** subcommands do not create directories, so any directories included in the *PathName* must already exist before using this subcommand. Do not include the home directory as part of the path name. If not defined, copies of outgoing mail are not saved.

screen=*Number*

If defined, controls how many lines of the message list are displayed at a time. You can set this option to show a certain number of lines on the screen. Each message in your mailbox has a one-line heading in the message list. If you have more than 24 messages, the first headings from the message list scroll past the top of your screen whenever you display the message list.

SHELL=*PathName*

Path name of the shell to use in the **!** command and the **~!** command. Your default shell is used if this option is not defined.

toplines=*Number*

If defined, gives the number of lines of a message to be displayed with the **top** subcommand; normally, the first five lines are printed.

VISUAL=*PathName*

Defines the editor that you start with the **~v** and **visual** subcommands. The value of *PathName* must be the full path name to the editor program that you want to use. The default is **/usr/bin/vi**.

Creating Aliases and Distribution Lists

If you send mail on a large network or often send the same message to a large number of people, entering long addresses for each receiver can become tedious. To simplify this process, you can create an alias or a distribution list in your **$HOME/.mailrc** file.

alias — A name that you define that can be used in place of a user address when you address mail.

distribution list — A name that you define that can be used in place of a group of user addresses when you address mail.

Aliases and distribution lists are used the same way and defined in similar ways; the only difference is the number of addresses defined for an alias (one address) and a distribution list (more than one address).

Related Information

Mail Overview for System Management on page 8–28 .
Message Handler (MH) Overview for System Management on page 9–1.
Network Overview on page 5–1.
How to send mail using the **mail** program on page 8–10.
How toUse the Mail Program subcommands to send mail on page 8–10.
How to Receive and Read Mail, using the **mail** program on page 8–12.
How to Forward Mail, using the **mail** program on page 8–14.
How to Reply to Mail, using the **mail** program on page 8–15.
How to Display the contents of a Mailbox, using the **mail** program on page 8–16.

How to Send Mail

Prerequisite Tasks or Conditions

1. The **mail** program installed on your system.
2. You must know the name and address of the person to send messages to.

Procedure

1. Enter the **mail** command on the command line followed by the name and address of the person, or persons, to receive the message:

   ```
   mail jan@brown
   ```

   The system responds with:

   ```
   Subject:
   ```

2. Type the subject of the message, and press Enter:

   ```
   Subject:  Dept Meeting
   ```

   When you have typed information in the subject field, and pressed Enter, you are ready to enter the body of the text.

3. Start typing your message (since the mail editor has the word wrap function, you don't have to press Enter at the end of each line).

   ```
   There will be a short department meeting this afternoon
   in my office. Please plan on attending.
   ```

4. Your cursor should be on a line by itself. Press the END OF TEXT character (Ctrl–d on most terminals). The system displays the carbon copy list:

   ```
   Cc:
   ```

5. Type the names and addresses for anyone you would like to send a copy of the message to, and press Enter. (If you don't want to copy anyone, press Enter.)

   ```
   Cc:  karen@hobo
   ```

6. When you finish entering the Cc: field information, your message is sent, and the system prompt is redisplayed.

How to Use the Mail Program Subcommands to Send Mail

1. To include a file in a message you are currently writing, enter:

   ```
   ~r schedule
   ```

   In this example, the information in the file `schedule` is included at the current end of the message being written. The `~r schedule` is entered at the left margin of the message.

2. To include a specific message in a message you are currently writing, enter:

   ```
   ~m 2
   ```

   In this example, message 2 is included at the current end of the message being written and is indented one tab from the left margin. This subcommand works only if you entered the mail editor from the mailbox prompt, using the **mail** subcommand, **reply** subcommand, or the **Reply** subcommand. The `~m 2` is entered at the left margin of the message.

3. To change the contents of the subject field, enter:

   ```
   ~s Vacation
   ```

   In this example, you change the subject field to Vacation. The **~s** subcommand is entered at the left margin of the message.

4. To quit the editor using the **~q** subcommand, enter:

   ```
   ~q
   ```

   The message is not sent. Instead, the message is saved in the **dead.letter** file in your home directory, unless you have not yet entered any text. You are returned to the operating system command line. The **~q** subcommand is entered at the left margin of the message.

5. To quit the editor using the interrupt key sequence, press:

   ```
   Ctrl–C
   ```

   The following message is displayed:

   ```
   (Interrupt -- one more to kill letter)
   ```

   Press Ctrl–C again.

   ```
   (Last Interrupt -- letter saved in dead.letter)
   ```

   The message is not sent. Instead, the message is saved in the **dead.letter** file in your home directory, unless you have not yet entered any text. You are returned to the operating system command line. If you press Ctrl–C before you press the Enter key after the first line of text, the message is not saved in the **dead.letter** file.

6. To add the contents of the **dead.letter** file into your current message, enter:

   ```
   ~d
   ```

   The contents of the **dead.letter** file is added to the current end of the message. You can then add to this message or simply send it. The **~d** subcommand is entered at the left margin of the message.

7. To invoke the alternate editor using the message text as the input file, enter:

   ```
   ~e
   ```

   You can now edit the message using the alternate editor instead of the mail editor. (The alternate editor can be defined with the **set EDITOR=** *editor* (where *editor* is the name of the editor you wish to use, for example, **vi**.) The ~e subcommand is entered at the left margin of the message.

   If you enter a name which is not known by the system or defined in an alias or distribution list, the system responds with the user ID followed by an error message:

   ```
   $ [user ID]...  User unknown
   ```

   Note: If you have personalized your system, your system prompt may be something other than $.

Related Information

Personalizing the Mail Program on page 8–5.
The **mail** Command.
The **mail** Command Editing Subcommands.

How to Receive and Read Mail

Prerequisite Tasks or Conditions

1. The **mail** program installed on your system.
2. Mail in your system mailbox.

Procedure

1. Enter the **mail** command:

   ```
   mail
   ```

   If there is no mail in your mailbox when you enter the **mail** command, the system responds with a message.

   If there is mail in your mailbox when you enter the **mail** command, the system displays a listing of the messages in your system mailbox.

2. Use these subcommands to display messages:

   a. To look at the next message in your mailbox, enter the following at the mailbox prompt (&):

      ```
      next
      ```

      The text for the next message is displayed. This message then becomes the current message. You can also press the Enter key to display the next message.

   b. To look at the contents of your personal mailbox, enter the following at the command line prompt:

      ```
      mail -f
      ```

      A listing of the messages in your personal mailbox, the **$HOME/mbox** file, is displayed.

   c. To look at the contents of a specific mail folder, enter the following at the command line prompt:

      ```
      mail -f +dept
      ```

      In this example, a listing of the messages in the `dept` folder is displayed.

   d. Use the **t** command to display the text of a particular message:

      ```
      t 3
      ```

   e. To look at several messages in your mailbox, enter the following at the mailbox prompt (&):

      ```
      type 2 3 4
      ```

      or

      ```
      type 2-4
      ```

In this example, the text for messages 2, 3, and 4 is displayed. Either the **type** subcommand or the **print** subcommand will display the text of the messages. When you display more than one message at a time, be sure to include the **set crt** statement in your **$HOME/.mailrc** file. You can also enter this subcommand at the mailbox prompt. If you do not use this subcommand and you have more than one screen of messages, the displayed messages scroll up and off the screen before you have time to read them.

3. Use this subcommand to delete messages:

   a. To delete the current message and display the next message, enter the following at the mailbox prompt (&):

      ```
      dp
      ```

      The current message is deleted and the next message displayed. You can also delete a specific message by entering `delete` followed by the message number.

4. Use the **q** command to exit the mailbox and return to the command line. The **mail** program saves the messages that you have read in your personal mailbox (**mbox**), unless you specify another action. If you did not read any of your mail, they will remain in the system mailbox until acted upon.

   Note: By default, the **mail** program stores messages you have read in your personal mailbox (**mbox**), unless you specify another file for your mail. Unread messages remain in your system mailbox (**/usr/spool/mail/*userID***) until you do something with them.

Related Information

Receiving and Handling Mail on page 8–2.

How to Forward Mail

Prerequisite Tasks or Conditions

1. The **mail** program installed on your system.
2. Mail in your system mailbox.

Procedure

To have your mail forwarded to an other person, perform the following steps:

1. Use the **cd** command with no parameters to ensure that you are in your home directory. The following command sequence illustrates that action for the user ID mary:

```
$  cd
$  pwd
/u/mary
$
```

2. While in your home directory, create a file called **.forward**. The **.forward** file contains the network address or addresses of the people that are to receive your forwarded network mail. This file must contain valid addresses. If it is a null file (zero length), your mail is not forwarded and is stored in your mailbox. You can use the **cat** command to create the .forward file as follows:

```
$  cat > .forward
mark
joe@sandwich
[END OF FILE]
$
```

Note: [END OF FILE] represents the END OF FILE character (Ctrl–D) on most terminals.) This must be entered on a line by itself.

In the preceding step, the **.forward** file contains the addresses of the users you want your mail forwarded to. Your mail will be forwarded to `mark` on your local system, and to `joe` on system `sandwich`.

3. To stop forwarding mail, using the **rm** command, remove the **.forward** file from your home directory:

```
$ rm .forward
```

In the preceding step, the **.forward** file was deleted using the **rm** command. Incoming mail is no longer being forwarded to other users.

Related Information

Receiving and Handling Mail on page 8–2.

How to Reply to Mail

Prerequisite Tasks or Conditions

1. The **mail** program installed on your system.
2. Mail in your system mailbox.

Procedure

1. To reply to a mail message, at the mailbox prompt (&), enter:

   ```
   r
   ```

   The **r** command creates a new message that is addressed to the sender of the selected message and copied to the people on the `Cc:` list (if there are any). The `Subject:` field of the new message refers to the selected message. The default value of the **r** command is the current message. This default can be overridden by entering the message number after the **r**:

2. To reply to a message, enter the following at the mailbox prompt (&):

   ```
   Reply 4
   ```

   The system responds with a message similar to the following:

   ```
   To: karen@thor
   Subject: Re: Department Meeting
   ```

   You then can enter your response:

   ```
   I'll be there.
   ```

   When you finish entering the text, press Ctrl–D to send the message. In this example, you are replying to message 4 sent by `karen@thor`. After the reply is sent, you are returned to the mailbox prompt.

3. To quit the mailbox, enter:

   ```
   q
   ```

   You have replied to the message, and have exited the **mail** program.

   Note: If you have personalized the **mail** program to meet your requirements, enter the character you selected to represent quit.

Related Information

Receiving and Handling Mail on page 8–2.

How to Display the Contents of a Mailbox

Prerequisite Tasks or Conditions

1. The **mail** program installed on your system.
2. Mail in your system mailbox.

Procedure

1. At the mailbox prompt (&), enter the **h** command to display the contents of your mailbox.

   ```
   & h
   ```

   Approximately 20 messages can be displayed at a time. The actual number displayed is determined by the type of terminal being used and the **set screen** command.

2. If you have more than 20 messages in your mailbox, at the mailbox prompt (&), enter:

   ```
   & h 21
   ```

   Message 21 and subsequent messages up to and including message 40 (if you have that number of messages in your mailbox) are displayed. Continue entering the **h** command until all messages have been displayed.

3. To return to the first group of 20 messages, at the mailbox prompt (&), enter:

   ```
   & h 1
   ```

   The first group of 20 messages is displayed.

Related Information

Receiving and Handling Mail on page 8–2.

How to Display Information about Selected Messages

Prerequisite Tasks or Conditions

1. The **mail** program installed on your system.
2. Mail in your system mailbox.

Procedure

1. To display heading information only for messages 1, 4, and 7, at the mailbox prompt (&), enter:

   ```
   & f 1 4 7
   ```

   Heading information for messages 1, 4, and 7 is displayed.

2. To display heading information only for messages 1 through 10, at the mailbox prompt (&), enter:

   ```
   & f 1-10
   ```

   Heading information for messages 1 through 10 is displayed.

3. To display messages received from address ron, at the mailbox prompt (&), enter:

   ```
   & f ron
   ```

   Messages, if any, from user `ron` are displayed. The characters entered for an address do not need to exactly match the address. They must only be contained in the address field of the messages in either uppercase or lowercase. Therefore, the request for address `ron` matches all of the following address (and many others): RoN, ron@topdog, hron, rOn.

4. To display messages that have meet in the `Subject:` field, at the mailbox prompt (&), enter:

   ```
   & f /meet
   ```

   Messages, if any, whose `Subject:` field contains the letters `meet` are displayed. The characters entered for a match pattern do not need to exactly match the `Subject:` field. They must only be contained in the Subject: field in either upper or lower case. Therefore, the request for subject `meet` matches all of the following subjects (and many others): Meeting on Thursday, Come to meeting tomorrow, MEET ME IN ST. LOUIS.

Related Information

Receiving and Handling Mail on page 8–2.

How to Create an Alias or Distribution List

Prerequisite Tasks or Conditions

1. The **mail** program installed on your system.
2. You must know the name and addresses of users you want to include in your alias or distribution list.

Procedure

1. To create an alias, enter the following statement in your **$HOME/.mailrc** file:

   ```
   alias kath kathleen@gtwn
   ```

   To send a message to user `kathleen` after you have added this line to your **$HOME/.mailrc** file, enter the following at the command line prompt:

   ```
   mail kath
   ```

   In this example, you create an alias for user `kathleen` whose userid and system name are `kathleen@gtwn`. You are now able to send mail to her using this alias.

2. To create a distribution list for your department, enter the following statement in your **$HOME/.mailrc** file:

   ```
   dept dee@merlin anne@anchor jerry@zeus bill carl
   ```

   To send a message to your department after you have added this line to your **$HOME/.mailrc** file, enter the following at the command line prompt:

   ```
   mail dept
   ```

   The message you now create and send will go to `dee` on system `merlin`, `anne` on system `anchor`, `jerry` on system `zeus`, and to `bill` and `carl` on the local system.

3. To list the aliases and distribution lists you have created, enter the following at the mailbox prompt (&):

   ```
   alias
   ```

   A list of the aliases and distribution lists is displayed.

Related Information

Receiving and Handling Mail on page 8–2.

How to Send a Vacation Message

Prerequisite Tasks or Conditions

1. The **mail** program installed on your system.
1. A **.forward** file must exist in your **$HOME** (login) directory, and must contain your user ID and address.
2. If you want to send a mail message, a **.vacation.msg** file must exist in your **$HOME** (login) directory. The **.vacation.msg** file contains your message.

Procedure

1. To create a vacation message, in your **$HOME** (login) directory, add the following line to the **.forward** file to initialize vacation:

   ```
   –I vacation
   ```

2. To create a vacation message, in your **$HOME** (login) directory, add the following line to the **.forward** file:

   ```
   \carl, "|vacation carl"
   ```

   The sender of the message will receive a message saying that Carl is on vacation.

3. To cancel the vacation message, remove the **.forward** file from your **$HOME** (login) directory:

   ```
   rm .forward
   ```

4. To send a message once a week to each person who has sent you mail, create the file **$HOME/.vacation.msg** and add your message to this file. This message will only be sent once a week to each unique sender. The following is an example of what this file might say:

   ```
   From: carl@odin.austin (Carl Jones)
   Subject: I am on vacation.

   I am on vacation until October 1. If you have something urgent, p
   lease contact Jim Terry <terry@zeus.valhalla>.
      —carl
   ```

   The above message will be sent once a week to each person who has sent you mail. If you have your mail forwarded to someone else, the message from the sender will also be forwarded to the person(s) defined in your **.forward** file.

5. To cancel the vacation message, remove the **.forward** file and .vacation.msg file from your **$HOME** (login) directory:

   ```
   rm .forward

   rm .vacation.msg
   ```

Related Information

How to Forward Mail on page 8–14.

How to Send and Receive Secret Mail

Prerequisite Tasks or Conditions

1. The **mail** program installed on your system.
2. A password must have been set up using the **enroll** command.

Procedure

1. To send secret mail to user `Barbara`, enter:

   ```
   xsend barbara
   ```

   The mail system is used to enter the text of the message. When you are finished entering your message, press Ctrl–D or a period (.) to exit the mail editor and send the message. The **xsend** command encrypts the message before it is sent.

2. To receive secret mail, enter:

   ```
   xget
   ```

   You are prompted for the password that was previously set up using the **enroll** command. After entering your password, the **xget** command prompt (&) is displayed and a listing of any secret mail is then displayed. The mail program is used to display any secret mail.

Related Information

How to Forward Mail on page 8–14.

How to Change Prompting for the Subject Field

Prerequisite Tasks or Conditions

1. The **mail** program installed on your system.

Procedure

1. To enable Subject Field prompting, edit the **.mailrc** file `ask` option:

   ```
   set ask
   ```

   Subject Field prompting is enabled.

2. To disable Subject Field prompting, edit the **.mailrc** file `ask` option:

   ```
   unset ask
   ```

   Subject Field prompting is disabled.

Related Information

Personalizing the Mail Program on page 8–5.
How to Change Prompting for the Carbon Copy (Cc:) Field on page 8–22.

How to Change Prompting for the Carbon Copy (Cc:) Field

Prerequisite Tasks or Conditions

1. The **mail** program installed on your system.

Procedure

1. To enable Carbon Copy (Cc:) Field prompting, edit the **.mailrc** file `askcc` option:

   ```
   set askcc
   ```

   Carbon Copy (Cc:) Field prompting is enabled.

2. To disable Carbon Copy (Cc:) Field prompting, edit the **.mailrc** file `askcc` option:

   ```
   unset askcc
   ```

   Carbon Copy (Cc:) Field prompting is disabled.

Related Information

Personalizing the Mail Program on page 8–5.
How to Change Prompting for the Subject Field on page 8–21.

How to Determine if Aliases or Distribution Lists are Defined

Prerequisite Tasks or Conditions

1. The **mail** program installed on your system.

Procedure

1. To display a listing of the aliases that are defined for the current mail session, at the mail prompt enter:

   a

   All aliases and distribution lists are displayed, one alias per line.

How to Change Text Editors Used for Entering Messages

Prerequisite Tasks or Conditions

- The **mail** program installed on your system.

Procedure

1. To change to the **e** editor, while in the **mail** program, enter:

   ```
   ~e
   ```

   This sequence activates the **e** editor, or other editor that you have defined in the **.mailrc** file. You can edit your mail message using this editor.

2. To change to the **vi** editor while in the **mail** program, enter:

   ```
   ~v
   ```

   This sequence activates the **vi** editor or other editor that you have defined in the **.mailrc** file. You can edit your mail message using this editor.

3. To define an editor to be used for entering mail messages, in your **.mailrc** file, enter:

   ```
   set EDITOR=pathname
   ```

   This command in your **.mailrc** file defines the editor that you activate with the ~e key sequence. The value of *pathname* must be the full pathname to the editor program that you want to use. The default is: **/usr/bin/e**. (set EDITOR=/bin/ed defines the **ed** editor for use with the ~e key sequence.)

   ```
   set VISUAL=pathname
   ```

   This command in your **.mailrc** file defines the editor that you activate with the ~v key sequence. The value of *pathname* must be the full pathname to the editor program that you want to use. The default is: **/usr/bin/vi**.

How to Save Mail

Prerequisite Tasks or Conditions

1. The **mail** program installed on your system.
2. Mail in your system mailbox.

Procedure

1. To save a message to a file, enter the following at the mailbox prompt (&):

   ```
   save 10 file1
   ```

   In this example, message number 10 is saved to the file `file1`. If the message number is not specified, the current message is saved.

2. To save a message to a folder, enter the following at the mailbox prompt (&):

   ```
   save 1 +procedures
   ```

   The following message is displayed:

   ```
   /u/jay/doc/procedures [Appended] 32/947
   ```

   In this example, message 1 has been added to the end of the folder `procedures`. Jay has the following **set folder** statement in his **$HOME/.mailrc** file so that the folder directory where that folder is kept is already selected:

   ```
   set folder=/u/jay/doc
   ```

3. To find the name of your current mailbox, enter the following at the mailbox prompt (&):

   ```
   folder
   ```

   If the current mailbox is `/u/lance/mbox`, the following is displayed:

   ```
   /u/lance/mbox: 2 messages 1 deleted
   ```

   This message indicates that `/u/lance/mbox` is the current mailbox, that it contains two messages, and that one of those messages will be deleted when you finish with this mailbox. Although the **mail** command displays the name of the current mailbox when it starts, you may lose track of what mailbox you are in.

Related Information

Mail Overview on page 8–1.
Using Mailboxes on page 8–5.

How to Personalize the Mail Program

Prerequisite Tasks or Conditions

- The **mail** program installed on your system.

Procedure

1. To see the **mail** options that are currently enabled, enter the following at the mailbox prompt (&):

   ```
   set
   ```

   A message similar to the following is displayed:

   ```
   &set
   ask
   metoo
   toplines 10
   &
   ```

   You can see from this list that two binary options are enabled: **ask** and **metoo**. Notice that there is no **askcc** entry in the list. This indicates that the **askcc** option is not enabled. You can also see that the **toplines** option has been assigned the value 10.

2. To prevent the date, from, and to headers from being displayed when a message is read with the **type** subcommand or the **print** subcommand, enter the following statement in your **$HOME/.mailrc** file:

   ```
   ignore date from to
   ```

   When a message is displayed using the **type** or **print** subcommand, the date, from, and to headers will not be displayed. However, if you want to display these headers without deleting the `ignore` statement from your **$HOME/.mailrc** file, use the **Type** subcommand, **Print** subcommand, or the **top** subcommand to display the message.

3. To get a list of the currently ignored header fields, enter the following at the mailbox prompt (&):

   ```
   ignore
   ```

   A list of all currently ignored headers is displayed.

   ```
   ask askcc
   ```

4. To prevent the **mail** banner from displaying when you start the **mail** program, enter the following statement in your **$HOME/.mailrc** file:

   ```
   set quiet
   ```

   The **mail** banner is the line at the top of the list of messages that shows the name of the **mail** program. It is similar to the following line:

   ```
   Mail [5.2 UCB]  [IBM AIX 3.1] Type ? for help.
   ```

5. To change the number of lines of the message list that are displayed at one time, enter the following statement in your **$HOME/.mailrc** file:

   ```
   set screen=20
   ```

In this example, the system will display 20 message headers at a time. Use the **headers** or **z** subcommands to view additional groups of headers. You can also enter this subcommand at the mailbox prompt.

6. To keep a record of messages you send to others, enter the following statements in your **$HOME/.mailrc** file:

```
set record=letters/mailout
set folder=letters
```

To read copies of the messages you have sent to others, enter the following:

```
mail -f +mailout
```

You can create new folders during a **mail** session, but a directory for storing them must exist before defining any new folders. In this example, the directory `$HOME/letters` must be created before messages are saved in it.

Related Information

Mail Overview on page 8–1.
Using Mailboxes on page 8–5.

Mail Overview for System Management

The **mail** command allows users to create and send messages either to other users on a network or to users on the local system. The **mail** command uses the **sendmail** command to route the message to its proper destination. The **sendmail** command reads the information in the **/usr/lib/sendmail.cf** file, the **/usr/lib/aliases** file, and the **/usr/lib/sendmail.nl** file to perform the proper routing.

The mail system is a general purpose, internetwork mail-routing facility. This system is not tied to any one transport protocol. It relays messages from one user to another across system and network domain boundaries. While processing the messages, the mail system can do a limited amount of message-header editing to put the message into a format that is appropriate for the receiving domain. With the **sendmail** command, you can log mail activities, control mail queues, and define mail aliases.

Related Information

The **mail** command, **sendmail** command.

How to Set Up Mail Delivery on page 8–62.

Message Handler (MH) Overview for System Management on page 9–1.

Understanding Mail Aliases

Aliasing maps names to address lists using a system-wide or personal file.

You can define two types of aliases: personal aliases and local-system aliases.

Personal aliases are produced by the individual user who defines the alias. In the **mail** program, these aliases are defined in the user's **$HOME/.mailrc** file.

Local system aliases apply to all mail handled by the **sendmail** command on the local system. Local system aliases rarely need to be changed. Define these aliases in the **/usr/lib/aliases** file. This is a text file that you can edit if you are a member of the system group. Anytime this file is edited, you must re-build the alias database using the **newaliases** command.

Related Information

The **newaliases** command.

The **sendmail** daemon.

The **/usr/lib/aliases** file.

Message Handler (MH) Overview for System Management on page 9–1.

How to Build the Alias Database on page 8–64, How to Create a Local System Alias on page 8–88.

Mail Overview for System Management on page 8–28.

Understanding the Mail Queue

The mail queue is used to store data and control files for mail messages that are queued prior to delivery. This might happen for several reasons. The most common are non-fatal errors in delivery caused by either a remote host not answering or a Basic Network Utilities (BNU) link to a long-distance number causing the mailer to be marked as *expensive*.

Entering the **sendmail** command with the **–q** flag and no time specified processes the queue once. Entering the **sendmail** command with the **–q** time flag processes the queue at intervals of time.When the **sendmail** command processes the queue, it does the following:

- Reads and sorts the queue.
- Processes unlocked jobs.

If a major system on your network is not operating, the queue at your system may become loaded with messages routed through that system. With a huge backlog of messages to deliver, performance on your system may go down. When the remote host is running again, **sendmail** automatically delivers the messages only if the **sendmail –q***time* command is running. If there is no time specified, you should run the **sendmail –q** command manually or through **cron**.

Related Information

The **sendmail** command.

How to Move the Mail Queue on page 8–69, How to Determine the Mail Queue Processing Interval on page 8–72, How to Examine the Message Queue Files on page 8–73, How to Flush the Mail Queue on page 8–71, How to Specify Time Values to sendmail on page 8–77.
Mail Overview for System Management on page 8–28.

Understanding Mail Logging Activities

Sendmail logs mail messages through the **syslogd** daemon. It logs to the file set up in **/etc/syslog.conf** file. The **syslogd** daemon must be configured and running before logging can occur. When activities occur that affect the mail system, the **sendmail** command writes messages into a log file. The default log file name is **log** in the mail queue directory (**/usr/spool/mqueue**, unless you change it). You must be a member of the system group to access this file.

Messages in the log file appear in the following format:

```
MMM DD HH:MM:SS hostname MMM DD HH:MM:SS sendmail [Pid]:
message_text
```

The symbols in this format have the following meanings:

MMM DD HH:MM:SS
: Specifies the time that the message was entered in the log file:

MMM	A 3-letter abbreviation for the month
DD	The date of the month (01 to 31)
HH	The hour of the day (00 to 23)
MM	The minute of the hour (00 to 59)
SS	The second of the minute (00 to 59).

PID
: This field contains the process ID of the process that generated the message.

Message Text
: Contains a brief explanation of the condition that caused the message to be written to the log file.

The date format is determined by the NLLDATE and NLTIME environment variables.

For example, the following entry occurred in a log file when the **sendmail** daemon was started:

```
Nov 16 11:00:29 lopnor sendmail [3926]
Daemon/queue proc started, Pid 3926
```

You can choose the types of activities that the **sendmail** command puts into the log file by changing the **L** option in the **/usr/lib/sendmail.cf** file. If you change the **/usr/lib/sendmail.cf** configuration file, you must process it with the **sendmail –bz** command before **sendmail** recognizes the change.

The level in the standard configuration file is level 9. It is set with the following statement that turns on the logging function:

```
OL9
```

The valid levels and their activities are as follows:

0
: Lists major activities only (building a configuration file or creating an alias database).

1
: Lists major problems only.

2
: Lists message collections and failed deliveries.

3	Lists successful deliveries.
4	Lists messages being deferred (due to a host being down, for example).
5	Lists placing messages in the queue (normal event).
6	Lists unusual but benign incidents (trying to process a locked file, for example).
9	Lists log internal queue IDs to external message IDs mappings. This can be useful for tracing a message as it travels among several hosts.
12	Lists several messages with information for debugging.
22	Lists verbose information regarding the queue and other activities.

Related Information

The **sendmail** command.

The **syslogd** daemon.

The **/usr/lib/sendmail.cf** file.

How to Display the Mailer Information on page 8–68, How to Log the Mailer Statistics on page 8–67, How to Manage the Log and Mail Queue on page 8–65.

National Language Support Overview in *General Programming Concepts*.

Mail Overview for System Management on page 8–28.

List of Directories and Files for the Mail System

The Mail system and the Message Handler package are user interfaces to the mail system. The following directories and files are associated with the **mail** program:

/usr/lib/Mail.rc A text file that you can modify to set the default characteristics of the **mail** command. This file sets local system defaults for all users of the Mail program.

$HOME/.mailrc A text file that each user creates in the **$HOME** directory. This file allows the user to change the defaults for the **mail** program when that user runs the program.

$HOME/mbox A text file that stores processed mail for the individual user.

/usr/bin/Mail, /usr/bin/mail, or /usr/bin/mailx
Three names linked to the same program. The **mail** program is one of the user interfaces to the mail system.

/usr/spool/mail The default mail drop directory. By default, all mail is delivered to the **/usr/spool/mail/UserName** file.

/bin/bellmail Performs local mail delivery.

/bin/rmail Remote mail interface for BNU. It is not the same **rmail** command that existed in earlier versions of the operating system.

Related Information

The **bellmail** command, **mail** command, **rmail** command.

Mail Overview for System Management on page 8–28.

List of the Directories and Files for the sendmail Command

Although the **sendmail** command can be used directly to send mail to other users, it is normally invoked only by the mail handling programs (Mail and Message Handler) to control mail routing. The **sendmail** command uses the following files. You can use different file names for many of these files by specifying a flag or configuration option to the **sendmail** command.

/usr/lib/sendmail The **sendmail** command.

/usr/ucb/mailq Links to the **/usr/lib/sendmail**. Using **mailq** is equivalent to executing the **/usr/lib/sendmail –bp** command.

/usr/lib/newaliases Links to the **/usr/lib/sendmail** file. Using **newaliases** is equivalent to executing the **/usr/lib/sendmail –bi** command.

/usr/lib/mailstats Formats and prints the **sendmail** statistics as found in the **/usr/lib/sendmail.st** file if it exists. The **/usr/lib/sendmail.st** file is the default, but you can specify an alternative file.

/usr/lib/edconfig Provides a menu interface for editing many parameters in the **sendmail** configuration file.

/usr/lib/aliases A text version of the aliases file for the **sendmail** command. You can edit this file to create, modify, or delete aliases for your system.

/usr/lib/aliasesDB A directory containing the aliases database files, **DB.dir** and **DB.pag**, that are created from the **/usr/lib/aliases** file when you run the **sendmail –bi** command.

/usr/lib/aliasesDBl A lock file for the **aliasesDB** database.

/usr/lib/sendmail.cf Contains the **sendmail** configuration information in text form. You can edit the file to change this information.

/usr/lib/sendmail.cfDB Contains the processed version of the **/usr/lib/sendmail.cf** configuration file. This file is created from the **/usr/lib/sendmail.cf** file when you run the **/usr/lib/sendmail –bz** command.

/usr/lib/sendmail.cfDBl A lock file for the **/usr/lib/sendmail.cfDB** database.

/usr/lib/sendmail.nl Contains the sendmail National Language Support (NLS) configuration information in text form. You can edit the file to change this information.

/usr/lib/sendmail.nlDB Contains the processed version of the **/usr/lib/sendmail.nl** configuration file. This file is created from the **/usr/lib/sendmail.nl** file when you run the **/usr/lib/sendmail –bn** command.

/usr/lib/sendmail.nlDBl A lock file for the **/usr/lib/sendmail.nlDB** database.

/usr/lib/smdemon.cleanu
A shell file that runs the mail queue and maintains the **sendmail** log files in the **/usr/spool/mqueue** directory.

/usr/lib/sendmail.st Collects statistics about mail traffic. This file does not grow. Use the **/usr/lib/mailstats** command to display the contents of this file. You may delete this file if you do not want to collect this information.

/usr/spool/mqueue Directory containing the temporary files associated with each message in the queue and may contain the log file.

/usr/spool/cron/crontabs
Directory containing files that the **cron** daemon reads to determine which jobs to start. The **root** file contains a line to start the **smdemon.cleanu** shell script.

Related Information

The **edconfig** command, **mailq** command, **mailstats** command, **newaliases** command, **sendmail** command.

The **syslogd** daemon.

Understanding the **sendmail.cf** File

National Language Support Overview in *General Programming Concepts*.

Mail Overview for System Management on page 8–28.

Editing the /usr/lib/sendmail.cf File with the edconfig Command

The **/usr/lib/sendmail.cf** file stores information about the type of mailer programs being used and information about how the **sendmail** command should format messages.

The **/usr/lib/sendmail.cf** file contains a set of information that allows the **sendmail** command to operate in the following environments with no changes:

- Local mail delivery
- Local area network delivery using TCP/IP
- Remote delivery using Basic Utilities Network (BNU).

If your environment includes any of these types of mail delivery, you can use the supplied configuration file with few, if any, changes.

Only on rare situations should you change this file.

The **/usr/lib/sendmail.cf** file contains control lines (host and domain name macros) that define the mail name of the local system. By default, the **sendmail** command reads the value returned by the **hostname** command to initialize the system's mail name. The configuration file should be changed only if the user wants the mail name of the system to be different from that set by the **hostname** command.

The **edconfig** command provides a menu interface for defining parameters in the **/usr/lib/sendmail.cf** configuration file. The **/usr/lib/sendmail.cf** configuration file must be in the format of the configuration file provided with the **sendmail** command. You must be a member of the system group to edit the **/usr/lib/sendmail.cf** file.

Related Information

The **edconfig** command, **hostname** command, **uname** command.

The **/usr/lib/sendmail.cf** file format.

How to Change Domain Name Macros on page 8–80, How to Change Host Name Macros on page 8–82.

How to Set Up Mail Delivery on page 8–62.

Building the **/usr/lib/sendmail.cf** Configuration File on page 8–37, Understanding the **sendmail.cf** File on page 8–50.

Mail Overview for System Management on page 8–28.

Building the /usr/lib/sendmail.cf Configuration File

To rebuild the database version of the configuration file each time you change the configuration file enter:

```
/usr/lib/sendmail -bz
```

This operation creates the file **/usr/lib/sendmail.cfDB**, which contains the database version of the configuration information. If the **sendmail** command is running as a daemon, you may instead issue a refresh command to it, which causes the daemon to rebuild the configuration database, the aliases database, and the NLS database. The daemon then rereads these databases so that it continues operation with the new data.

The syntax for the **refresh** command is as follows:

```
refresh -s sendmail
```

Or, to achieve the same effect, use the **kill** command as follows:

```
kill -1 'cat /etc/sendmail.pid'
```

This command sends a SIGHUP signal to the **sendmail** daemon, causing it to rebuild and reread the databases.

Related Information

The **kill** command, **refresh** command.

The **/usr/lib/sendmail.cf** file format on page 8–50.

Understanding the sendmail.cf File on page 8–50 and Specifying the File Revision Level on page 8–40.

Editing the **/usr/lib/sendmail.cf** File with the **edconfig** Command on page 8–36.

Understanding the NLS Configuration File

The **/usr/lib/sendmail.nl** file contains lists of systems that correctly interpret mail messages containing National Language Support (NLS) or ISO-8859/1 characters. If the recipient system of a message is not in either of these lists, all characters in the body of the message are changed to standard 7-bit ASCII characters.

If the recipient system is in the list of NLS systems, all NLS extended characters in the body of the message are converted to 7-bit *NCesc* escape sequences. These characters can then pass intact through intermediate mail systems that may strip the eighth bit from all messages. The receiving AIX mail system then can convert these characters back to extended NLS characters before depositing the mail in the user's mail box.

If the recipient system is in the list of ISO-8859/1 systems, each NLS extended character in the body of the mail message is converted to the closest ISO-8859/1 character. This character set is used by some computer systems.

Each item in the NLS and ISO-8859/1 lists is a Regular Expression (RE). For each non-local recipient, the **sendmail** command compares the recipient's address with the names in the **/usr/lib/sendmail.nl** file after first passing the address through rule set 33. In the default **/usr/lib/sendmail.cf** file, rule set 33 causes the user and some types of route information to be stripped from the address. Rule set 33 normally parses from the right side until an ! (exclamation point) or an @ (at sign) character is encountered. If an ! (exclamation point) is encountered, all characters to the right are stripped. If an @ (at sign) is encountered, all characters to the left are stripped. The **sendmail** command rule set 33 distinguishes between uucp-style and domain-style addresses. The REs match the ultimate system destination.

After the address has been stripped using rule set 33, the result is compared to each regular expression in the compiled version of the **/usr/lib/sendmail.nl** file until a successful match occurs.

The following is an example of the **/usr/lib/sendmail.nl** file:

```
#list the nls compatible systems
nls:    ^@.*madrid\.,
        ^@.*italy\.europe$,
        lisbon!,
        !nagaski!$,
        berlin

#list the ISO-8859 compatible systems
8859:   ^vienna!,
        ^@bangok\.thailand,
        ^@tangiers,
        ^@kinshasa
```

The example `^@.*madrid\.,` matches any address which has this name as a subdomain. Although this may work in most cases, it could match other unintended destinations. The `^@.*italy\.europe$,` line is a more precise example of this type which matches all domains under this domain. The `lisbon!` example matches any uucp-addressed mail item destined for that system or passing through it. Because the `!nagasaki!$,` example is anchored on the right, it matches all users on that system. The `berlin` example is an RE, which matches any address that has this character string anywhere in it. The remaining examples show how a list of ISO-8859/1 systems might look. Notice that the period . (period)

in domain-style addresses must be typed as . (period) Escape so that it refers to an actual dot rather than the period, which has special meaning in REs.

Related Information

The **sendmail** command.

The **/usr/lib/sendmail.nl** file

Mail Overview for System Management on page 8–28.

National Language Support Overview in *General Programming Concepts*.

Specifying the File Revision Level

The configuration file revision-level macro, **Z**, helps to track changes that you make to the configuration file. Each time you make a change to the configuration file, also change the value of this macro. Choose any format for the number that you define. For example, if the configuration file is at level 3.1, the following entry should be in the configuration file:

```
DZ3.1
```

You can also use a text string for this macro. For example:

```
DZversion_one
```

Scroll forward to Understanding sendmail.

Related Information

The **/usr/lib/sendmail.cf** file format.
Understanding the **sendmail.cf** File on page 8–37.
Editing the **/usr/lib/sendmail.cf** File with the **edconfig** Command on page 8–36.

Understanding sendmail

The Mail program and the Message Handler program are user interfaces to the mail system. You can create or manipulate messages with these programs and use the **sendmail** command to route the messages. Then, the **sendmail** command uses other mailers to actually send the mail. The following figure is an example of a common sendmail configuration.

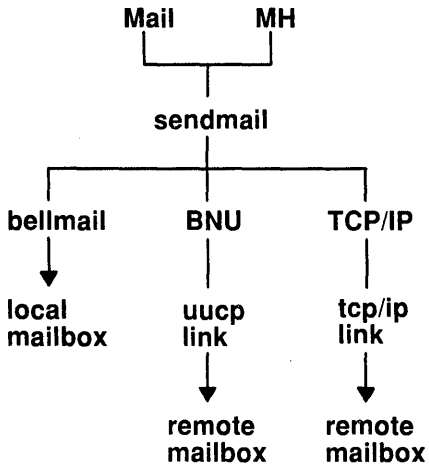


Figure 1. Example of most common sendmail configuration

To deliver local mail, the **sendmail** command routes the message through the **bellmail** command. The **bellmail** command sends the message on the local system only.

To deliver mail over a network link, the **sendmail** command routes the message through a Basic Network Utilities (BNU) link.

To deliver TCP/IP-routed mail, the **sendmail** command establishes a TCP/IP link to the remote system. The Simple Mail Transfer Protocol (SMTP) is then used to transfer the message data to the remote system.

Related Information

Basic Networking Utilities Overview in *Communications Programming Concepts.*

The **bellmail** command, **sendmail** command.

Mail Overview for System Management on page 8–28.

Message Handler (MH) Overview for System Management in *Communication Concepts and Procedures.*

List of Mail Management Commands

Choose the command you want to read about:

comsat Notifies users of incoming mail (daemon).

edconfig Edits values in a **sendmail** configuration file.

mailq Prints the contents of the mail queue.

mailstats Displays statistics about mail traffic.

newaliases Builds a new copy of the alias database from the **/usr/lib/aliases** file.

rmail Handles remote mail received through the **uucp** command of the Basic Networking Utilities.

sendmail Routes mail for local or network delivery.

smdemon.cleanu
Cleans up the **sendmail** queue for periodic housekeeping.

Understanding System–Defined Macros

Note: This section is intended for users who have an advanced understanding of the Mail program.

Warning: Altering system-defined macros may render the Mail Program unusable.

Some macros are defined internally for use in writing stanzas in the configuration file. You should not define these macros in the configuration file, but you can use the symbols when writing other macros and parsing rules by priority in the configuration file. The following macros are system-defined:

- Date Macros
- Message-Routing Macros
- Macros That Identify the Recipient
- Macros That Identify the Sender
- Required Macros.

Related Information

Message Routing Macros on page 8–47, Macros That Identify the Recipient on page 8–46, Macros That Identify the Sender on page 8–45, Required Macros on page 8–48.

Mail Overview for System Management on page 8–28.

Date Macros

Warning: Altering date macros may render the Mail Program unusable.

Three macros that represent dates:

a Date extracted from the `DATE:` line of the message (if there is one). If no `DATE:` line is found in the incoming message, `$a` is set to the date in this format as follows:

```
Fri, 9 Oct 87 09:25:47 CDT
```

b Used for postmarks. It is the current date in this format as follows:

```
Fri, 9 Oct 87 09:25:47 CDT
```

d Current date in **ctime** format, as follows:

```
Fri Oct  9 09:25 CDT 1987
```

Related Information

Message Routing Macros on page 8–47, Macros That Identify the Recipient on page 8–46, Macros That Identify the Sender on page 8–45, Required Macros on page 8–48.

Understanding System-Defined Macros on page 8–43.

Macros That Identify the Sender

Warning: Altering macros that identify the sender may render the Mail Program unusable.

The following macros identify the sender of the message:

f Contains the sender's address.

g Contains the sender's address relative to the recipient.

s Contains the sender's host name.

x Contains the full name of the sender. This name is determined in one of the following ways:

- Passed as a flag to the **sendmail** daemon.
- Value of the `Full-name:` line in the message heading.
- Comment field of a `From:` line.
- If the message originated locally, the full name can be looked up in the **/etc/passwd** file.

y Contains the ID of the sender. For outgoing mail, you can specify the sender's full name with the `Full-name:` header field, the **-F** command-line flag, or the *NAME* environment variable. If none of these are defined, the **sendmail** daemon tries to get the sender's full name from the additional data field of the **/etc/passwd** file entry for the sender. For incoming mail, the sender's full name is obtained from the message's `From:` or `Full-name:` header lines.

Related Information

Date Macros on page 8–44, Macros That Identify the Recipient on line 8–46, Message–Routing Macros on page 8–47, and Required Macros on page 8–48.

The **sendmail** daemon.

Understanding System–Defined Macros on page 8–43.

Macros That Identify the Recipient

Warning: Altering macros that identify the recipient may render the Mail Program unusable.

The following macros identify the recipient of the message:

h Contains the name of the receiving host (as set by the `$@` part of the mailer resolution line in Rule Set 0).

u Contains the ID of receiving user (as set by the `$:` part of the mailer resolution line in Rule Set 0).

z Contains the home directory of the receiving user if the message is being delivered locally.

Related Information

Date Macros on page 8–44, Message-Routing Macros on page 8–47, Macros That Identify the Sender on page 8–45, Required Macros on page 8–48.

Understanding System–Defined Macros on page 8–43.

Message–Routing Macros

Warning: Altering message-routing macros may render the Mail Program unusable.

Macros to track and route messages through the network:

c Contains the hop count. The hop count is the number of times that this message has been processed and usually corresponds to the number of separate time stamps in the message. (Some mail handlers allow you to change the hop count.)

i Contains the mail queue ID of the message on this host. When put into the time stamp line of the message, this macro is useful for tracking messages.

p Contains the process ID of the **sendmail** daemon. To create unique strings for the `Message–ID:` field in the message, the **sendmail** command uses this macro, along with the **$t** macro.

r Contains the protocol used to communicate with the **sendmail** daemon.

t Contains a numeric representation of the current time. To create unique strings for the `Message–ID:` field in the message, the **sendmail** daemon uses this macro, along with the **$p** macro.

v Contains the version name and number of the AIX Operating System running on the system.

w Contains the name of the local host.

Related Information

Date Macros on page 8–44, Macros That Identify the Recipient on page 8–46, Macros That Identify the Sender on page 8–45, Required Macros on page 8–48.

The **sendmail** daemon.

Understanding System-Defined Macros on page 8–43.

Required Macros

Warning: Altering required macros may render the Mail Program unusable.

The **sendmail.cf** configuration file that comes with the **sendmail** command defines the following macros:

e Denotes the Simple Mail Transfer Protocol (SMTP) entry message.

j Denotes the official domain name for this site.

l Denotes the format of the UNIX From line (not the `From:` line).

n Denotes the name of the daemon (for error messages).

o Denotes the set of operators in addresses.

q Denotes the default format of sender address.

If you create a new configuration file, be sure to include definitions for these macros.

Related Information

Date Macros on page 8–44, Macros That Identify the Recipient on page 8–46, Macros That Identify the Sender on page 8–45, Message Routing Macros on page 8–47, Understanding System–Defined Macros on page 8–43.

The **sendmail** daemon.

Mail Overview for System Management on page 8–28.

Understanding Rewrite Rules

Note: This section is intended for users with an advanced understanding of the Mail program.

The **sendmail** command receives addresses in a number of different formats and changes them to the format needed to route the message for the mailer program being used. To perform this translation, the **sendmail** command uses a set of rewrite rules that are defined in the **/usr/lib/sendmail.cf** configuration file.

When the **sendmail** command processes an address, it tries to match the pattern of that address format to a pattern specified by the left side of one of the rewrite rules in the **/usr/lib/sendmail.cf** configuration file. When it finds a rule that matches, it changes the format of the incoming address to the format specified by the right side of that rewrite rule.

The **/usr/lib/sendmail.cf** configuration file that is installed with the **sendmail** command contains enough rules to perform the translation for BNU and TCP/IP networks using a domain address structure. You should not have to change these rules unless you are connecting to a system that uses a different addressing scheme.

Related Information

The **sendmail.cf** configuration file.

The **sendmail** command.

Mail Overview for System Management on page 8–28.

Understanding the sendmail.cf File

Note: This section is intended for users who have an advanced understanding of the Mail topic.

The **/usr/lib/sendmail.cf** configuration file contains the configuration information for the **sendmail** command. After changing any information in the **/usr/lib/sendmail.cf** file, use the **sendmail –bz** command. This command compiles the file into a database format that the **sendmail** command can read.

The **/usr/lib/sendmail.cf** configuration file consists of a series of control lines, each of which begins with a single character that defines how the rest of the line is used. Lines beginning with a space or a tab are continuation lines. Blank lines and lines beginning with a # (pound sign) are comments. Control lines are used for the following functions:

- Defining macros and classes for use within the configuration file
- Defining message headings
- Defining a mailer
- Defining message precedence
- Defining administrative IDs
- Setting options for the **sendmail** command.

You can edit the **/usr/lib/sendmail.cf** configuration file either using the **edconfig** command or using a text editor. Editing the file with the **edconfig** command provides a menu interface to the configuration file. The **edconfig** command allows you to edit the most commonly needed configuration options.

In most cases, you do not need to change any of the parameters to get your mail system working, but to make changes to the **/usr/lib/sendmail.cf** file other than those permitted by the **edconfig** command, you must use a text editor. Most of these parameters should not be changed casually, since they affect how the **sendmail** command interfaces with the system and networks.

Warning: The tab character may be defined as the field-separator character in rule sets when the **sendmail** command reads the file. Some editors store tabs as the number of spaces they represent instead of the tab character itself. If you edit the configuration file with one of these editors, the configuration file will not be built successfully. You can solve this problem by using the vi editor or by changing the field-separator character with the **J** option.

Defining Macros and Classes (D Control Line, C Control Line, and F Control Line)

Macros and classes in the **/usr/lib/sendmail.cf** configuration file are interpreted by the **sendmail** command. A macro is a symbol that represents a value or string. A macro is defined by a D control line in the **/usr/lib/sendmail.cf** file. Macros are not expanded until the **sendmail** command loads the rule sets when it starts up.

A class is a symbol that represents a set of one or more words such as a file name. Classes are used in pattern matching when the **sendmail** command is parsing addresses. You can create a class using a list or you can create a class using a file.

To use a macro or class in a control line, put a $ (dollar sign) before its name. For example, if the name of the macro is **x**, use **$x** when using that macro in a control line. Without the preceding **$**, the **sendmail** command interprets **x** as only the letter x. To specify conditional expressions, use the following format:

`$?`*`MacroText1`*`$|`*`Text2`*`$.`

In this format, the symbols have the following meaning:

`$?`	If. (If this symbol is used, a clause must follow.)
Macro	The macro being tested.
Text1	The pattern to be used if **$x** macro is defined.
`$|`	Else. (This symbol is optional. If it is used, the Text 2 expression must follow.)
Text2	The pattern to be used if **$***Macro* is not defined.
`$.`	Specifies the end of the conditional expression.

Note: Do not use any of the characters defined as tokens (by the required **o** macro) when defining a word in a class. The **sendmail** command may not be able to read the definition correctly.

Defining Message Headings (H Control Line)

Lines in the configuration file that begin with a capital letter `H` define the format of the headers used in messages. The format of the `H` control line is:

`H[?`*`MailerFlags`*`?]`*`FieldName`*`: `*`Content`*

Here, the variable parameters have the following meanings:

MailerFlags	This field is optional. If you supply it, surround it with ? (question marks). This field contains mailer flags that determine whether the `H` line is used. If the mailer being used requires the field defined by this control line (as indicated in the mailer definition's flags field), then the `H` control line is included when formatting the heading. Otherwise, the `H` control line is ignored.
FieldName	This field contains the text that is displayed as the name of the field in the heading information. Some typical field names include `From:`, `To:`, and `Subject:`.
Content	This field defines the information that is displayed following the field name. Usually macros specify this information.

The following example lines are from a typical **/usr/lib/sendmail.cf** file:

```
H?P?Return-Path: <$g>
```

This line defines a field called `Return-Path` that displays the content of the **$g** macro (sender address relative to the recipient). The `?P?` portion indicates that this line is only used if the mailer uses the **P** flag (the mailer requires a `Return-Path` line). The header will be automatically generated only if the mailer has the indicated flag; but if the header appears in the input message, it will be passed through unchanged.

```
HReceived: $?sfrom $s $.by $j ($v/$Z)
        id $i; $b
```

This line defines a field called `Received`. This field displays the following information:

`$?sfrom $s $.`	Displays the text `from` followed by the content of the **$s** macro if an **s** macro is defined (sender's host name).
`by $j`	Displays the text `by` followed by the content of the **$j** macro (official name for this site).
`($v/$Z)`	Displays the version of the **sendmail** command (**$v**) and the version of the **/usr/lib/sendmail.cf** file (**$Z**), set off by parentheses and separated by a slash.
`id $i;`	Displays the text `id` followed by the content of the **$i** macro (mail-queue ID of the message) and a ; (semicolon).
`$b`	Displays the current date.

Defining a Mailer (M Control Line)

Note: Defining a mailer in the **/usr/lib/sendmail.cf** configuration file does not ensure that it will be used. You must also define rewrite rules to ensure an address format resolves to that mailer.

A mailer is a program that delivers mail either locally or over some type of network to another system. Use control lines that begin with the letter `M` to define the characteristics of a mailer that interfaces with the **sendmail** command. The format of a mailer definition control line is as follows:

```
M=MailerName, P=Path, F=Flags, \
S=Sender, R=Recipient, E=EndOfLine, \
A=Argument, M=Max
```

The following paragraphs and examples describe the parameters for the mailer definition.

Specifying a Mailer Name (MMailerName)

Each mailer must have an internal name. The name can be any string that you choose, except that the names `local` and `prog` are reserved. You must provide definitions for these two mailers in the **/usr/lib/sendmail.cf** configuration file if they are not already there (the default configuration file contains these definitions). To define the mailer name, put the name immediately after the `M` in the mailer definition control line:

```
MMailerName
```

For example, the following mailer-definition fragment introduces the definition line for a mailer called `lan`:

```
Mlan
```

Defining the Path to the Mailer (P=Path)

Specify the location of the mailer with the `P` field in the mailer definition. This field has the format:

`P=`*`Path`*

Here, the *`Path`* variable defines the full path name of the mailer on the local system. If the mailer uses the Simple Mail Transfer Protocol (SMTP), use the `[IPC]` string as the path. For example, the following two mailer definition fragments define a local mailer at `/bin/bellmail` and another mailer that uses SMTP:

```
Mlocal, P=/bin/bellmail,
Mlan, P=[IPC],
```

Specifying Mailer Flags (F=Flags)

Flags provide further information to the **sendmail** command about the mailer being described. Specify mailer flags with the `F` field in the mailer definition. This field has the format:

`F=`*`Flags`*

This field defines the meaning for the flags that the **sendmail** command recognizes. For example, the following mailer-definition fragment uses the **rlsm** flags to indicate that the mailer requires a **–r** flag, delivers locally, needs quotation marks stripped from addresses, and can deliver to more than one user at a time:

```
Mlocal, P=/usr/bin/mail, F=rlsm,
```

You can also define mailer flags to match flags that you define in special header definitions in your **/usr/lib/sendmail** configuration file.

Specifying the Rewrite Rule Sets for the Mailer (S=Sender R=Recipient)

The **sendmail** command uses sets of rewrite rules to change the format of incoming addresses to a style that the receiving mailer can understand. With the `S` field, specify the rewrite rule set to use on sender addresses for this mailer. With the `R` field, specify the rewrite rule set to use on recipient addresses for this mailer. These fields have the following format:

`S=`*`Sender`*`, R=`*`Recipient`*

Here, *`Sender`* specifies a particular rule set number for processing addresses for this mailer.

Defining a Different End-of-Line String (E=EndOfLine)

The normal indication of the end of line is a string that contains only the new-line character. To change this string, use the optional `E` field. The format of this field is:

`E=`*`EndOfLine`*

Here, *`EndOfLine`* is the character string that specifies the end of the line. You can use normal \ (backslash) escape characters to specify the end-of-line string (\r, \n, \f, \b).

Passing Information to the Mailer (A=Argument)

Specify information to be passed to the mailer with the optional `A` field. This field has the following format:

```
A=Argument
```

Here, *`Argument`* can be any string of words with imbedded spaces allowed. Any or all of the words can be macros, such as the **$u** macro (receiving user name). If you do not include this field, or the field does not contain the **$u** macro, the **sendmail** command uses the SMTP to send messages to the mailer. If the `P` field for this mailer (access path name) is the `[IPC]` string, indicating a mailer accessed with interprocess communications, use the following information in this field:

```
A=IPC $h [Port]
```

Here, *`Port`* is the optional port number to which to connect.

Limiting Message Size (M=Max)

Use the optional **M=***Max* mailer flag to specify the maximum size in bytes of messages that the mailer handles. For example, the following field specifies a maximum limit of 10,000 bytes:

```
M=10000
```

Example Mailer Specifications

To Specify a Local Delivery Mailer:

```
Mlocal, P=/bin/bellmail, F=lsDFMmn, S=10, R=20, A=mail $u
```

The mailer is called `local` and its path name is **/bin/bellmail**. The mailer takes the following flags:

l	Specifies local delivery.
s	Strips quotation marks from addresses.
DFM	Needs `Date:`, `From:`, and `Message-ID:` fields.
m	Delivers to multiple users.
n	Does not need an operating system `From` line at the start of the message.

Rule Set 10 should be applied to sender addresses in the message, and Rule Set 20 should be applied to recipient addresses. Additional information sent to the mailer in the `A` field is the word `mail` and words containing the recipient's name.

To Specify a Mailer for Local Area Network Delivery:

```
Mlan, P=[IPC], F=meC, S=11, R=21, A=IPC $h, M=100000
```

The mailer is called `lan` and connected through an interprocess communication link. Messages are sent to it using the SMTP mailer. This mailer can handle multiple users at once (`m`), the connection is defined as expensive (`e`), and any domain from the sender address should be appended to any recipient name without a domain (`C`). Sender addresses should be processed by Rule Set 11 and recipient addresses by Rule Set 21. There is a 100 000 byte limit on messages passed through this mailer (`M=100000`).

Setting Options for the sendmail Command (OOption[Value])

You can set configuration options for use by the **sendmail** command by using a control line in the configuration file. The options that can be set are the same as those you can specify with the **–o** flag to the **sendmail** command. An option is named with a single character. The format of the set-option control line is:

O*Option*[*Value*]

Here, *Option* is a single-character name for the option being set and *Value* is either a string, an integer, a time interval, or a Boolean option. Legal values for a Boolean option are `t,T,y,Y` for a true value and `f` or `F` for a false value. If you do not specify a value for a Boolean option, it becomes a true value.

For example, the following entries from the default **/usr/lib/sendmail.cf** file show the format of the set-option control line:

OA/usr/lib/aliases Sets the option `A` variable to the full path name of the aliases file (**/usr/lib/aliases**).

OL9 Sets the log-level option `L` variable to a value of 9. This entry occurs early in the **/usr/lib/sendmail.cf** file to ensure that the log is maintained during the reading of the file.

Oo Sets a Boolean to true. (Sets Boolean option `o` to true.)

Or5m Sets the read `time out` option r to a value of 5 minutes.

Setting Delivery Mode (OdValue)

The **sendmail** command can operate in several delivery modes. The default configuration file sets the delivery mode to `b` (the default value). However, you can change the delivery mode with the `Od`*Value* option in the configuration file. The Od delivery mode option has the format:

`Od`*Value*

These modes specify how promptly mail is delivered. Legal values include the following:

i Delivers interactively.

b Delivers in background. (This is the default.)

q Queues the message and delivers during queue run.

Name Server Options (OOption[Value])

The sendmail command can use a name server to resolve recipient addresses. The following options determine whether and how it uses the name server:

I Treats a failure to connect to the name server as a temporary error. The message is queued and delivery can be retried later.

K[Value] Sets the types of name server resource records that the **sendmail** command uses to resolve recipient addresses. Possible values are:

MB	Uses Mailbox (MB) records to resolve recipient user names.
MG	Uses Mail Group (MG) records to resolve recipient user names.
MR	Uses Mail Rename (MR) records to resolve recipient user names.
MX	Uses Mail Exchanger (MX) records to resolve recipient host names.
ALL	Uses all of the above records.

If Value is not specified, the name server is not used to resolve recipient addresses.

Note: You can use any combination of these values in a **sendmail** command. If using more than one K*Value* option, separate them by a space.

NLS Options (OOption)

The following options determine how the **sendmail** command handles mail messages that contain National Language Support (NLS) or ISO-8859/1 characters.

k Prevents character-set conversions for outgoing messages. Any NLS extended characters are sent to other systems intact.

lFile Sets the NLS configuration file to File.

w Causes all incoming mail that has eight-bit characters to be treated as ISO-8859/1 mail. These characters are converted to the equivalent NLS extended characters.

Time Out Options (OrTimeValue and OTTimeValue)

The **sendmail** command can time out when reading standard input or when reading from a remote SMTP process. The default configuration file sets this value to 5 minutes. This value should be correct for most situations. However, if you need to change the time-out value, change the `r` option in the configuration file. The `r` option has the format:

`Or`*`TimeValue`*

Here, *`TimeValue`* is the time unit that the **sendmail** command should wait until timing out.

After sitting in the queue for a period of time, the **sendmail** command notifies the sender of the message that it could not be sent. The default time out is set to 3 days. To set this time out with the `T` option in the **/usr/lib/sendmail.cf** configuration file enter:

`OT`*`TimeValue`*

Here, *`TimeValue`* is the time unit the **sendmail** command leaves the message in the queue before timing out the message.

Operational Logging Level Option (OLNumber)

The `OLNumber` option specifies the log level to be used when the **sendmail** command is running.

The format of the line to change the operational logging level is:

`OLNumber`

Following is a list of valid levels (*Number*) and the activities that they represent. Each number includes the activities of all numbers of lesser value and adds the activity they represent:

0	Logs major activities only (building a configuration file or creating an alias database).
1	Logs major problems only.
2	Logs message collections and failed deliveries.
3	Logs successful deliveries.
4	Logs deferred messages (for example, a host being down).
5	Logs messages that are placed in the queue (normal event).
6	Logs unusual but benign incidents (for example, trying to process a locked file).
9	Logs the internal queue ID to external message ID mappings. This can be useful for tracing a message as it travels between several hosts. (This is the default.)
12	Logs several messages that are of interest when debugging.
22	Logs verbose information regarding the queue and other activities.

Message Queue Options (OOption [Value])

Three options adjust the way the **sendmail** command handles the message queue.

The following *Option* values are allowed:

c	Causes the **sendmail** command to queue messages without sending them if an outgoing mailer is marked as expensive to use. The queue can be run when costs are lower or when the queue is large enough to send the message efficiently.
Q*Directory*	Sets the directory in which to queue messages. The directory will be created if it does not exist.
s	Enqueues messages before delivery, even when in immediate delivery mode.
Y	The **sendmail** command delivers each message in the mail queue from a separate process. This option is not required and can increase overhead in the AIX environment.

Mail Statistics File Option (OSFile)

You can specify the path name of the mail statistics file, in which the **sendmail** command stores data about delivered and received messages.

S*File* — Sets the mail statistics to the file specified by the *File* variable. Statistics are only collected if the file exists. This file must be created by the user.

Returned Mail Option (OPAddress)

You can specify someone to receive copies of all returned mail.

P*Address* — Identifies the person to receive a copy of all returned mail.

Alias Options (OOption[Value])

A*File* — Uses the file specified by the *File* variable as the alias file.

m — Sends messages to the sender if the sender appears in an alias expansion of a recipient address. The default action is to remove the sender's address from recipient alias expansions.

n — Validates the right-hand side of alias definitions when performing the **newaliases** function.

Header Options (OOption[Value])

f — Saves `From` lines at the front of messages. These lines are normally discarded.

o — Indicates that this message can have old style headers. Without this option, the message has new style headers (commas instead of spaces between addresses). If this option is set, an adaptive algorithm correctly determines the header format in most cases.

Verbose Mode Option (Ov)

v — Runs in verbose mode.

ID Options (OOptionNumber)

The following options allow you to set the default group and user IDs for mailers. The **sendmail** command sets the process group and user IDs to these values before invoking the mailer program:

`O`*`OptionNumber`*

The following options are available to set group and user IDs:

g*Number* — Sets the default group ID to the value specified by the *Number* variable. The **sendmail** command uses this ID when it invokes mailers.

u*Number* — Sets the default user ID to the value specified by the *Number* variable. The **sendmail** command uses this ID when it invokes mailers.

Character Interpretation Options (OOption[Value])

The **sendmail** command interprets certain characters as having a special function. Use the format:

O*Option*[*Value*]

You can set the interpretations with the following options:

B*Character* Sets the blank substitution character to the *Character* parameter. The **sendmail** command replaces spaces without quotes in addresses with the character specified by the *Character* variable. The supplied configuration file uses a . (dot) for the value of the *Character* variable.

i Does not interpret a . (dot) on a line by itself as a message terminator.

Note: Enabling this feature will prohibit the **sendmail** command from receiving messages through SMTP (since the **sendmail** command cannot recognize the end of the message when receiving messages through SMTP).

Rewrite Rules Options (OOption[Value])

The following two options allow you to alter the **sendmail** command's use of rewrite rules. Use the format:

O*Option*[*Value*]

The following option is available to the **sendmail** command for rewrite rules:

J*Character* Allows spaces as well as tabs to separate the left-hand side and right-hand side of rewrite rules. For both the left-hand side and right-hand side, the *Character* variable must be used in place of embedded spaces. The default for the *Character* variable is _ (underscore). All instances of the *Character* variable are changed to spaces after the left-hand side and right-hand side are separated by the **sendmail** command. This option allows rewrite rules to be modified using an editor that replaces tabs with spaces.

Error-Processing Modes Option (OeValue)

The following option sets the error-processing mode. Use the format:

Oe*Value*

The following options are available to the **sendmail** command for error processing modes:

Valid modes are:

e Mails the error message to the user's mail box, but always exits with a zero exit status (normal return).

m Mails the error message to the user's mail box.

p Displays the error message on the terminal (default).

q Discards the error message and returns the exit status only.

w Writes the error message to the terminal if delivering in interactive mode; otherwise it mails the error message to the user's mail box.

Macro Definition Option (OMacroValue)

You can use the O control line to define a macro. Use the following option to do so:

M*Macro Value* Defines the *Macro* variable to have the value specified by the *Value* variable. This option is normally used only from the **sendmail** command line.

Configuration File Revision-Level Option (DZNumber)

The configuration file revision-level **Z** macro helps you track changes that you make to the **/usr/lib/sendmail** configuration file. Each time you make a change to the **/usr/lib/sendmail** configuration file, you should also change the value of this macro. Choose any format for the number you define. For example, if the **/usr/lib/sendmail** configuration file is at level 3.1, the following entry should be in the **/usr/lib/sendmail** configuration file:

```
DZ3.1
```

You can also use a text string for this macro. For example:

```
DZversion_one
```

Defining Message Precedence (P Control Line)

The **/usr/lib/sendmail.cf** configuration file contains lines to define mail-queue precedence for messages that contain a `Precedence:` field. Normally, you do not need to change the values in the default **/usr/lib/sendmail.cf** configuration file.

The name defined and the numerical value assigned are based on the needs of the network. Higher numbers have higher priority. Numbers less than 0 (zero) indicate that error messages will not be returned to the sender of these messages. The `Precedence:` field value is 0 for any precedence name not defined in the configuration file. For example, the configuration file may contain the following entries:

```
Pfirst-class=0
Pspecial-delivery=100
Pbulk=-60
Pjunk=-100
```

These entries set `special-delivery` as the highest priority message and `junk` as the lowest priority.

Defining Administrative IDs (T Control Line)

Administrative IDs can override the sender address using the **-f** flag to the **sendmail** command. The **/usr/lib/sendmail.cf** configuration file defines these IDs with the `T` control line. For example, the configuration file may contain the following entries:

```
Troot
Tdaemon
Tuucp
```

These entries define `root`, `daemon`, and `uucp` user IDs as administrative IDs for the **sendmail** command.

These IDs can also be defined using only one `T` control line:

```
Troot daemon uucp network
```

Files

/usr/lib/sendmail.cf	Contains the configuration file for the **sendmail** command.
/usr/lib/sendmail.cfDB	Contains compiled version of the **/usr/lib/sendmail** configuration file.

Related Information

The **edconfig** command, **sendmail** command.

The Mail sendmail.cf File.

Understanding Rewrite Rules on page 8–49, Understanding System-Defined Macros on page 8–43.

Editing the **/usr/lib/sendmail.cf** File with the **edconfig** Command on page 8–36.

Mail Overview for System Management on page 8–28.

How to Set Up Mail Delivery

Prerequisite Task

You must be a member of the system group.

Procedure

The **sendmail** command is now part of the the AIX Base Operating System (BOS) and is installed with BOS.

1. Ensure that the configuration information in the configuration file (**/usr/lib/sendmail.cf**) accurately reflects your system's needs.
2. Ensure that the alias information in the **/usr/lib/aliases** file is accurately set up and that this reflects your system's needs.
3. If TCP/IP is installed, uncomment sendmail's **startsrc** command in the **/etc/rc.tcpip** file. This ensures that the **sendmail** daemon will be started at sysem boot time.

Related Information

The **sendmail** command.

The **/usr/lib/sendmail.cf** configuration file.

How to Build the Alias Database on page 8–64, How to Define the Addressing and Routing Information for Mail on page 8–63.

Understanding Mail Aliases on page8–29 .

Mail Overview for System Management on page 8–28.

How to Define the Addressing and Routing Information for Mail

Prerequisite Tasks

None.

Procedure

- To deliver mail to users on the local system, no additional addressing and routing information is required. The **sendmail** command routes local mail into a users' system mailbox in the /usr/spool/mail directory.
- To deliver mail in a local area network, the network must be using the TCP/IP network protocol. To simplify the task of keeping the address information up-to-date, you may want to set up a name server for the network. A name server controls the addressing information for the network and for connections to systems beyond the immediate network.
- To deliver mail to another system connected to the local system with a UUCP link, the link must be defined to the Basic Networking Utilities in the BNU configuration files. The Basic Networking Utilities system is the version of UUCP used in AIX. The **sendmail** command transfers mail to the BNU commands for delivery across a UUCP link.

Related Information

BNU Overview for System Management in *Communications Programming Concepts.*

The **sendmail** command.

Mail Overview for System Management on page 8–28.

How to Build the Alias Database

Prerequisite Tasks

The **/usr/lib/aliases** file must contain the required aliases, which are as follows:

MAILER–DAEMON
The ID of the user who is to receive messages addressed to the mailer daemon. This name is initially assigned to the root user:

```
MAILER–DAEMON: root
```

postmaster The ID of the user responsible for the operation of the local mail system. The **postmaster** alias defines a single mailbox address that is valid at each system in a network. This address allows users to send inquiries to the **postmaster** alias at any system without knowing the correct address of any user at that system. This name is initially assigned to the root user:

```
postmaster: root
```

nobody The ID that is to receive messages directed to programs such as **news** and **msgs**. This name is initially assigned to **/dev/null**:

```
nobody: /dev/null
```

If you wish to receive these messages, define this alias to be a valid user.

Procedure

The **sendmail** command does not use the alias definitions in the local-system **/usr/lib/aliases** file directly. Instead, the **sendmail** command reads a processed database manager (dbm) version of the **/usr/lib/aliases** file.

1. Use the following command to process the local-system **/usr/lib/aliases** file so the **sendmail** command can read it:

   ```
   /usr/lib/sendmail –bi
   ```

2. This command causes the **sendmail** command to read the local-system **/usr/lib/aliases** file and create two additional files containing the alias database information. These new alias files are:

   /usr/lib/aliasesDB/DB.dir

   /usr/lib/aliasesDB/DB.pag

Note: If these files do not exist, the **sendmail** command cannot start and generates an error message when trying to start.

Related Information

How to Create a Local System Alias on page 8–88.

The**/usr/lib/aliases** file.

The **sendmail** command.

Understanding Mail Aliases on page 8–29.

How to Manage the Log and the Mail Queue

Prerequisite Tasks

None.

Procedure

Because information is continually appended to the end of the log file, it can become very large. Also, error conditions can cause unexpected entries to the mail queue. To keep the mail queue and log from growing too large, use the **smdemon.cleanu** command in the **/usr/lib** file path. This command forces the **sendmail** command to process the queue, and also maintains four progressively older copies of log files, named **log.0**, **log.1**, **log.2**, and **log.3**. Each time the command runs it moves:

1. **log.2** to **log.3**
2. **log.1** to **log.2**
3. **log.0** to **log.1**
4. **log** to **log.0**.

This action removes the **/usr/spool/mqueue.log** file, allowing logging to start over with a new file. You can change this command to create more or fewer old copies of the log file. This program can be invoked manually or run at a specified interval by the **cron** daemon. To have the **cron** daemon start this command at a specified time, use the following procedure:

1. Log in as root user and then change to the **crontabs** directory by entering the following:

   ```
   cd /usr/spool/cron/crontabs
   ```

2. Edit the **root** file.
3. The file contains a line similar to the following line:

   ```
   #45 23 * * * ulimit 5000; /usr/lib/smdemon.cleanu > /dev/null
   ```

   Remove the # (pound sign) at the beginning of that line. This entry runs the **/usr/lib/smdemon.cleanu** shell script every day at 11:45 PM. If your system is not operating at that time, change the second number (`23` representing the 23rd hour, or 11:00 p.m.) to a time when your system is on. For example, to run the cleanup daemon at 4:45 p.m. enter:

   ```
   45 16 * * * ulimit 5000; /usr/lib/smdemon.cleanu > /dev/null
   ```

4. Save the file and exit from your editor.
5. Notify the **cron** daemon of the change made to the file by reloading the file while still in the **crontabs** directory:

   ```
   crontab root
   ```

Related Information

The **sendmail** command, **smdemon. cleanu** command.

The **cron** daemon.

The **sendmail.cf** file format.

How to Display the Mailer Information on page 8–68, How to Log the Mailer Statistics on page 8–67.

Understanding the Mail Logging Activities on page 8–31.

How to Log Mailer Statistics

Prerequisite Task

Must be a member of the system group before you can create the **/usr/lib/sendmail.st** file.

Procedure

The **sendmail** command can keep track of the volume of mail being handled by each of the mailer programs that interface with it (those mailers defined in the **/usr/lib/sendmail.cf** file).

1. To start the accumulation of mailer statistics, create the **/usr/lib/sendmail.st** file by entering the following:

   ```
   touch /usr/lib/sendmail.st
   chmod 660 /usr/lib/sendmail.st
   ```

   When this file exists and is either empty or contains a correctly formatted data structure, the **sendmail** command keeps track of the mail volume handled by each mailer in a database in that file.

   If the **sendmail** command encounters errors when it tries to record statistics information, the command writes a message through the **syslog** subroutine. These errors do not affect other operations of the **sendmail** command.

   The **sendmail** command updates the information in the file each time that it processes mail. The size of the file does not grow, but the numbers in the database do. They represent the mail volume since the time you created or reset the **/usr/lib/sendmail.st** file.

2. To reset the data and start counting over again, specify the **–z** flag to the **mailstats** command as follows:

   ```
   /usr/lib/mailstats –z
   ```

Related Information

The **mailstats** command, **sendmail** command.

The **syslog**, **openlog**, **closelog**, or **setlogmask** subroutine.

The **/usr/lib/sendmail.cf** file format.

How to Display Mailer Information on page 8–68, How to Manage the Log and the Mail Queue on page 8–65.

Understanding Mail Logging Activities on page 8–31.

How to Display Mailer Information

Prerequisite Tasks or Conditions

None.

Procedure

The information kept in the **/usr/lib/sendmail.st** file is in a database format that cannot be read as a simple text file.

1. To display the mailer statistics, use the **mailstats** command:

   ```
   /usr/lib/mailstats
   ```

   This command reads the information in the **/usr/libsendmail.st** file, formats it, and then writes it to standard output. The format of the information is shown in the following example:

   ```
   Mailer      msgs_from       bytes_from      msgs_to bytes_to
   local               1                2            1      201
   ```

   The fields in the report have the following meanings:

   msgs_from Contains the number of messages received by the local machine from the indicated mailer.

   bytes_from Contains the number of bytes in the messages received by the local machine from the indicated mailer.

   msgs_to Contains the number of messages sent from the local machine using the indicated mailer.

   bytes_to Contains the number of bytes in the messages sent from the local machine using the indicated mailer.

   If the **sendmail** command transmits mail directly to a file, such as the **$HOME/dead.letter** file or an alias target, the message and byte counts are credited to the prog mailer.

Related Information

The **mailstats** command, **sendmail** command.

How to Log Mailer Statistics on page 8–67, How to Manage the Log and the Mail Queue on page 8–65.

Understanding Mail Logging Activities on page 8–31.

How to Move the Mail Queue

Prerequisite Tasks or Conditions

None.

Procedure

A major host that goes down for an extended period usually results in many entries in the mail queue. As a result, the **sendmail** command spends a long time sorting the queue. If you move the queue to a temporary place and create a new queue, the old queue can be run later when the host returns to service. Use the following procedure to move the queue to a temporary place and create a new queue:

1. If the **sendmail** daemon was started by SRC, shut it down by entering:

   ```
   stopsrc -s sendmail
   ```

   If this command does not stop the daemon, or if it was not started by SRC, enter:

   ```
   kill 'cat /etc/sendmail.pid'
   ```

   If this command still does not stop the daemon, enter:

   ```
   kill -9 'cat /etc/sendmail.pid'
   ```

2. To move the entire queue directory, use the following commands:

   ```
   cd /usr/spool
   mv mqueue omqueue
   ```

3. To restart the **sendmail** daemon, enter:

   ```
   startsrc -s sendmail -a "-bd [-qQPI]"
   ```

   OR

   ```
   /usr/lib/sendmail -bd -q[QPI]
   ```

   The `QPI` is optional. It is the value of the queue-processing interval; for example, 30 million.

4. Use the following command to run the old mail queue:

   ```
   /usr/lib/sendmail -oQ/usr/spool/omqueue -q
   ```

   The **-oQ** flag specifies an alternate queue directory, and the **-q** flag specifies to run every job in the queue. To get a report about the progress of the operation, use the **-v** flag. (This operation can take a long time.)

5. Remove the log files and the temporary directory when the queue is empty and after you have processed the old log files according to your local procedures (saving them to another directory or to diskette). To remove these files, enter the following commands:

   ```
   rm /usr/spool/omqueue/*
   rmdir /usr/spool/omqueue
   ```

Related Information

The **sendmail** daemon.

The **kill** command, **startsrc** command, **stopsrc** command.

How to Determine the Mail Queue Processing Interval on page 8–72, How to Examine the Message Mail Queue Files on page 8–73, How to Examine the q File on page 8–75, How to Flush the Mail Queue on page 8–71, How to Specify Time Values to the Sendmail command on page 8–77.

Understanding the Mail Queue on page 8–30.

How to Flush the Mail Queue

Prerequisite Tasks or Conditions

None.

Procedure

Each message in the mail queue contains a time value that indicates the time of submission. As a result, you can flush messages that have been in the queue for a given period by running the queue with a shorter message time out.

The following command runs the queue, flushes any messages that are more than one day old (–oT1d), and tries to send all messages in the queue that have not exceeded the time-out value:

```
/usr/lib/sendmail –oT1d –q
```

Related Information

The **sendmail** daemon.

How to Move the Mail Queue on page 8–69, How to Determine the Queue-Processing Interval on page 8–72, How to Examine the Message Queue Files on page 8–73, How to Examine the q File on page 8–75, How to Specify Time Values to Sendmail Command on page 8–77.

Understanding the Mail Queue on page 8–30.

How to Determine the Queue–Processing Interval

Prerequisite Condition

You must be a member of the system group.

Procedure

The interval at which the **sendmail** daemon processes the mail queue is determined by the value of the **–q** flag when the daemon starts. The **/etc/rc.tcpip** shell file contains a default value for the queue-processing interval of 30 minutes.

If this value does not suit your system, you can change it as follows:

1. Edit the **/etc/rc.tcpip** file.
2. Find a line that assigns a value to the *qpi* variable, such as:

   ```
   qpi=30m
   ```

   This value indicates a queue-processing interval of 30 minutes.
3. Change the value assigned to the *qpi* variable to the time value that suits your needs.

   These changes will take effect the next time the system is restarted. If you want the changes invoked immediately, you must:

   a. Stop the **sendmail** daemon.

      ```
      stopsrc –s sendmail
      ```

   b. Restart the **sendmail** daemon.

      ```
      startsrc –s sendmail –a "–bd –qQPI"
      ```

Related Information

The **sendmail** command, **startsrc** command, **stopsrc** command.

The **/etc/rc.tcpip** file.

TCP/IP Overview on page , How to Examine the Message Queue Files on page 8–73, How to Examine the q File on page 8–75, How to Flush the Mail Queue on page 8–71, How to Move the Mail Queue on page 8–69, How to Specify Time Values to Sendmail on page 8–77.

Understanding the Mail Queue on page 8–30.

How to Examine the Message Queue Files

Prerequisite Condition

You must be a member of the system group to work with this directory.

Procedure

The **sendmail** command keeps message queue files in a spool (**/usr/spool/mqueue**) directory by default. Each message in the message queue has a number of files in this directory. The files are named in the following format:

TypeLetterID

Here, *ID* is the message queue ID, *Letter* is the fixed letter `f`, and *Type* can be one of the following letters indicating the type of file:

d The data file containing the message body without the heading information.

l A lock file. This file may not be present. If it is, the job is currently being processed and a queue run will not process the file. For that reason, an extraneous **l** file can cause a job to apparently disappear without timing out.

n File created when an ID is being created. Because this file is present for only a short time, you may not see an **n** file unless an error occurs. It is a separate file to ensure that no mail can ever be destroyed due to a race condition.

q The queue-control file. This file contains the information necessary to process the job.

t A temporary file. This file is an image of the **q** file when it is being rebuilt. It is quickly renamed to the **q** file.

x A transcript file that exists during the life of a session and shows everything that happens during that session.

If a message has a queue ID of AA00269, the following files are created and deleted in the mail queue directory while the **sendmail** command tries to deliver the message:

dfAA00269 Data file

lfAA00269 Lock file

nfAA00269 Backup file

qfAA00269 Control file

tfAA00269 Temporary file

xfAA00269 Transcript file.

Related Information

The **sendmail** command.

The **/usr/spool/mqueue** Directory.

How to Determine the Queue Processing Interval on page 8–72, How to Examine the q File on line 8–75, How to Flush the Mail Queue on page 8–71, How to Move the Mail Queue on page 8–69.

Understanding the Mail Queue on page 8–30.

How to Examine the q File

Prerequisite Tasks or Conditions

None.

Procedure

The **q** file contains a series of lines each beginning with a code letter. The code letters are as follows:

D Contains the name of the data file. There can be only one of these lines.

H Contains a heading definition. There may be any number of these lines. The order in which the `H` lines appear determines their order in the final message. These lines use the same syntax as heading definitions in the **/usr/lib/sendmail.cf** configuration file.

M Contains a message printed by the **mailq** command. It is mainly used to store status information.

P Contains the priority of the current message. The priority is used to order the queue. Higher numbers mean lower priorities. The priority increases as the message sits in the queue. The initial priority depends on the message class and the size of the message.

R Contains a receiver address. There is one line for each receiver.

S Contains the sender address. There is only one of these lines.

T Contains the message creation time used to compute when to time out the message.

The following is a sample **q** file sent to `amy@zeus`:

```
P217031
T566755281
DdfAA00269
MDeferred: Connection timed out during user open with zeus
Sgeo
Ramy@zeus
H?P?return-path: <geo>
Hreceived: by george (0.13 (NL support)/0.01)
        id AA00269; Thu, 17 Dec 87 10:01:21 CST
H?D?date: Thu, 17 Dec 87 10:01:21 CST
H?F?From: geo
H?M?message-id: <8712171601.AA00269@george>
HTo: amy@zeus
Hsubject: test
```

This example shows:

`P217031` Priority of the message.

`T566755281` Submission time in seconds.

`DdfAA00269` Name of the data file.

`MDeferred: Connection timed out during user open with zeus` Status message.

`Sgeo` ID of the sender.

`Ramy@zeus` ID of the receiver.

Related Information

The **sendmail** command, **mailq** command.

The **sendmail.cf** file.

How to Determine the Queue–Processing Interval on page 8–72 , How to Examine the Message Queue Files on page 8–73., How to Flush the Mail Queue on page 8–71, How to Move the Mail Queue on page 8–69, and How to Specify Time Values to Sendmail on page 8–77.

Understanding the Mail Queue on page 8–30.

How to Specify Time Values to Sendmail

Prerequisite Tasks or Conditions

None.

Procedure

All time intervals are set using `Number[Unit]`, where `Number` is an integer value and `Unit` is the unit letter. You can specify an arbitrary number of intervals, which are added together to compute the total interval. If a `Unit` is not speciied, the **sendmail** daemon uses days as the default unit. For example, `10` represents 10 days. The full set of scales is:

s Seconds

m Minutes

h Hours

d Days

w Weeks.

Related Information

The **sendmail** daemon.

How to Determine the Queue–Processing Interval on page 8–72, How to Examine the Message Mail Queue Files on page 8–73, How to Examine the q File on page 8–75, How to Flush the Mail Queue on page 8–71, How to Move the Mail Queue on page 8–69.

Understanding the Mail Queue on page 8–77.

How to Edit the sendmail.cf File with the edconfig Command

Prerequisite Conditions

You must be a member of the system group to edit the **/usr/lib/sendmail.cf** file with this or any other program.

Procedure

The **edconfig** command found in the **/usr/lib/edconfig** file path provides a menu interface for defining some of the parameters in the configuration file (**sendmail.cf**). Use this command to edit the local configuration file or to set up a configuration file to be used on another system. The configuration file must be in the same format as the **/usr/lib/sendmail.cf** file format.

Use the following procedure to edit the **/usr/lib/sendmail.cf** configuration file with this program:

1. To change to the **/usr/lib** directory, enter:

   ```
   cd /usr/lib
   ```

2. To start the **edconfig** editor with the configuration file, enter:

   ```
   /usr/lib/edconfig sendmail.cf
   ```

   The **edconfig** program starts and displays a menu.

3. Follow the instructions on the screen to find and change the desired configuration parameters.

4. Exit the program by entering the number found on the menu for writing the configuration file and exiting. If you do not want to keep the changes that you made, enter the number for exiting without writing to the configuration file.

The **edconfig** command allows you to define or change the following types of entries in the configuration file:

- The content of the *HostName* class and *HostName* macro
- The *DomainName* macro
- The four macros that define the separate tokens of the *DomainName* parts
- Configuration options (with help information) for:
  - Operational logging level (OL option)
  - Default delivery mode (Od option)
  - Alias file path (OA option)
  - Statistics file path (OS option)
  - Queue directory path (OQ option)
  - Maximum message retention time in queue (OT option)
  - Queueing uses of expensive mailers (Oc option)
- Configuration file revision level.

Related Information

The **edconfig** command.

The **sendmail.cf** file format.

Building the **/usr/lib/sendmail.cf** Configuration File on page 8–37.

Understanding the sendmail.cf File on page 8–50.

How to Change the Domain Name Macro

Note: This section is intended for the user with an advanced understanding of the Mail program.

Warning: Altering the macro files may render the Mail Program unusable.

Prerequisite Tasks

None.

Procedure

The domain name macro, **DD**, specifies the full domain name of your local group of hosts. This macro can be automatically set through the **hostname** command.

Sendmail will read what has been set with the **hostname** command and use it to initialize the host and domain macros and classes. The configuration file macros only need to be changed if the user wants the sendmail host and domain names to be different from that set by the **hostname** command.

Note: Only change the domain name macros if you want to override the default. The default is the domain name returned by the **gethostname** command.

The domain name is a series of names separated by periods (.) in the following format:

name1.name2.name3.name4

Specify the domains if they exist. You do not need to have all of the parts listed. For example, the following are all properly formatted domain name macro definitions:

```
DDpub.aus.ibm.com
or
DDrch.ibm.com
or
DDacct.xyz
```

Changing the Domain Name Part Macros

The domain name part macros specify the individual parts of the full domain name. The values that you enter for each of the domain name part macros must be the same as the corresponding parts that you specify for the full domain name. If you use less than four parts for your full domain name, always begin with the **DE** macro and define as many other macros in order as needed.

The following macros define the four parts of the domain name:

DE Defines the first part of the domain name. Place the lowest level of your domain name directly following the letters **DE**:

DE*LowestLevel*

DF Defines the second part of the domain name. Place the next level of your domain name directly following the letters **DF**:

DF*NextLevel*

DG Defines the third part of the domain name. Place the next level of your domain name directly following the letters **DG**:

```
DGNextLevel
```

DH Defines the fourth part of the domain name. Place the next level of your domain name directly following the letters **DH**:

```
DHNextLevel
```

Related Information

The **hostname** command.

How to Change the Host Name Class on page 8–83, How to Change the Host Name Macro on page 8–82, How to Change the **sendmail.cf** File Revision-Level Macro on page 8–84.

Building the **/usr/lib/sendmail.cf** Configuration File on page 8–37.

How to Change the Host Name Macro

Note: This article is intended for users with an advanced understanding of the Mail program.

Warning: Altering the host name macro may render the Mail Program unusable.

Prerequisite Tasks or Conditions

None.

Procedure

The host name macro, **Dw**, specifies the name of your host system that is used in the return address of all messages that you generate.

Sendmail will read what has been set with the **hostname** command and use it to initialize the host and domain name macros and classes. The configuration file macros only need to be changed if the user wants the **sendmail** command host and domain names to be different from that set by the **hostname** command.

Note: Only change the host name macros if you want to override the default. The default is the host name returned by the **gethostname** subroutine.

Start the **edconfig** command and edit the host name macro to match the name of your host system. Use the following format:

```
Dwhostname
```

In this format, *hostname* is the name of your host system. For example, the following entry defines the host name for system `george`.

```
Dwgeorge
```

Note: If the **Dw** macro is defined, you must also define the **CW** class.

Related Information

The **hostname** command, **sendmail** command.

The **gethostname** subroutine.

How to Change the Domain Name Macro on page 8–80, How to Change the Host Name Class on page 8–83, How to Change the sendmail.cf File Revision-Level Macro on page 8–84.

Mail Overview for System Management on page 8–28 , Understanding the sendmail.cf File on page 8–50.

How to Change the Host Name Class

Prerequisite Tasks or Conditions

None.

Procedure

The **sendmail** command will read what has been set with the **hostname** command and use it to initialize the host and domain name macros and classes. The configuration file macros only need to be changed if the user wants the sendmail host and domain names to be different from that set by the **hostname** command.

The host name class, **Cw**, specifies the name and all aliases for your host system. If your system uses different names for two different network connections, enter both names as part of the host name class. If you do not define both names, mail sent to the undefined name is returned to the sender.

1. Determine the aliases that your system uses.
2. Start the **edconfig** command and edit the host name class to match these aliases. Use the following format:

   ```
   Cwhostname alias1 ... aliasn
   ```

   In this format, *hostname* is the name of your host system and *alias1* through *aliasn* are alternate names by which your system is known. For example, the following entry defines the host name class for system `george`.

   ```
   Cwgeorge george2
   ```

Related Information

The **hostname** command, **sendmail** command, **edconfig** command.

The **sendmail.cf** file.

How to Change the Domain Name Macro on page 8–80, How to Change the Host Name Macro on page 8–82, How to Change the sendmail.cf File Revision-Level Macro on page 8–84.

Understanding the sendmail.cf File on page 8–50.

How to Change the sendmail.cf File Revision–Level Macro

Note: This section is intended for users with an advanced understanding of the Mail program.

Prerequisite Tasks or Conditions

None.

Procedure

The configuration file revision-level macro, **Z**, helps you track changes that you make to the configuration file. Each time you make a change to the configuration file, also change the value of this macro. You can choose any format for the number that you define. For example, if the configuration file is at level 3.1, the following entry should be in the configuration file:

```
DZ3.1
```

You can also use the following text string for this macro:

```
DZversion_one
```

Related Information

The **hostname** command.

The **sendmail.cf** file.

How to Change the Domain Name Macro on page 8–83, How to Change the Host Name Class on page 8–83, How to Change the Host Name Macro on page 8–82.

Mail Overview for System Management on page 8–28, Understanding the sendmail.cf File on page 8–50.

How to Create a Macro

Note: This section is intended for a user with an advanced understanding of the Mail program.

Prerequisite Tasks or Conditions

None.

Procedure

Use the control line that begins with the letter `D` to define a macro. The syntax for the **D** macro definition is:

`DxVAL`

In this format *x* is the name of the macro and *VAL* is the value assigned to it.

For example, the following control line in the configuration file defines a macro `D`, representing the name of the local domain, to be the name assigned to that domain.

In the following example, the local domain is assigned the name `802`, but the name could be any alphanumeric string that you or your network administration chooses.

```
DD802
```

To use a macro elsewhere in the **/usr/lib/sendmail.cf** file, precede it with a `$` (dollar sign) character. The **sendmail** command will replace the `$x` with the value of the `x` macro.

The following example shows the use of a previously defined macro to define a new macro. It defines a required `j` macro, representing the official name for the local site, to be the value of the `$w` macro (the host name of the local site), as follows:

```
Dj$w
```

Related Information

How to Create a Class Using a File on page 8–86, How to Create a Class Using a List on page 8–87.

The **sendmail** command.

Macros That Identify the Sender on page 8–45.

Mail Overview for System Management on page 8–28.

How to Create a Class Using a File

Prerequisite Tasks or Conditions

None.

Procedure

Use a control line that begins with the letter `F` to define a class whose members are listed in an external file. The syntax for the `F` class definition is:

```
FClass File [Format]
```

In this format, *Class* is the name of the class that matches any of the words listed in *File*. The format reads the elements of the class (X) from *FileName* using an optional **scanf** subroutine format specifier. The format specifier can contain only one conversion specification. Only one class member is read for each line in *FileName*.

Related Information

How to Create a Class Using a List on page 8–87, How to Create a Macro on page 8–85

The **scanf** subroutine.

Mail Overview for System Management on page 8–28.

How to Create a Class Using a List

Prerequisite Tasks or Conditions

None.

Procedure

Use a control line that begins with the letter `C` to define a class with a specified list of members. The syntax for the `C` class definition is:

```
CClass Word1 Word2 ...
```

In this format *Class* is the name of the class that matches any of the named words; for example, `Word1` and `Word2`. No word can include characters defined as delimiters, such as a . (period). You can spread the assignment of words to the class across many lines. For example, the control line:

```
CDLOCAL 802
```

defines the `D` class to represent the class of names `LOCAL` and `802`. You can accomplish the same function with two separate control lines:

```
CDLOCAL
CD802
```

A class is used in the left side of rewrite rules to match components of addresses. The expression `$=x` matches any word in class `x`. The expression `$~x` matches any word *not* in class `x`.

Related Information

The **sendmail.cf** file.

How to Create a Class Using a File on page 8–86, How to Create a Macro on page 8–85.

Understanding the **sendmail.cf** File on page 8–50, Understanding Rewrite Rules on page 8–49.

Mail Overview for System Management on page 8–28.

How to Create a Local System Aliases for Mail

Prerequisite Conditions

To create a new local-system alias, you must be a member of the system group.

Procedure

1. Edit the **/usr/lib/aliases** file. Aliases in this file are defined with an entry in the following format:

   ANname: *Name1*, *Name2*, ... *NameX*

   AName can be any alphanumeric string that you choose (not including special characters, such as @ or !). *Name1* through *NameX* is a series of one or more addresses.

2. Continue the list of names on one or more lines. Begin each continued line with a space or a tab. Blank lines and lines beginning with a # (pound sign) are comment lines that are not included in the processed local-system **/usr/lib/aliases** file.

   For example, the following entry defines the `writers` alias to be a set of addresses of people in that group:

   ```
   writers: geo, mark@zeus, ctw@athena, brian
   ```

   This definition could also be contained on several lines, as long as each added line begins with a space or a tab as follows:

   ```
   writers: geo,
         mark@zeus,
         ctw@athena,
         brian
   ```

3. Process the **/usr/lib/aliases** file to produce an alias database by using the **/usr/lib/sendmail –bi** command.

4. Create an owner for any distribution list aliases. If the **sendmail** command has trouble sending mail to the distribution list, it sends an error message to the owner of that list. For example, the following set of entries in the **/usr/lib/aliases** file defines a distribution list named `editors`, whose owner is `glenda@hera`:

   ```
   editors: glenda@hera, davidm@kronos, perryw@athena
   owner–editors: glenda@hera
   ```

Related Information

The **sendmail** command.

The **/usr/lib/aliases** file.

How to Build the Alias Database on page 8–64.

Understanding Mail Aliases on page 8–29.

Chapter 9. Message Handler

The Message Handler (MH) package enables you to create, distribute, receive, view, process, and store messages. This chapter contains information to help you manage, customize, and use the MH package.

Message Handler (MH) Overview for System Management

The Message Handler (MH) package contains a collection of commands that enable you to create, distribute, receive, view, process, and store messages. You can run the MH commands as separate programs, use MH commands with other MH and AIX commands, and create programs that call MH commands.

The MH package uses the AIX file and directory system. MH commands store messages as individual files with numerical file names, and group messages into directories. A directory of messages is called a folder. In addition to actual messages, each folder may contain a number of standard files to support the MH functions.

Since the messages are AIX files and the folders are AIX directories, you can use the AIX commands to manage messages and folders. For example, you can use the **chmod** command to change the access permissions on a folder, or you can run the **cron** command to purge old or inactive messages from a group of folders.

The MH package does not provide a message transport facility. MH commands provide input to the **sendmail** command and rely on the transport facilities associated with the **sendmail** command.

Related Information

The **sendmail** command.

The MH Alias file format.

The **.mh_profile** file.

Mail Overview for System Management on page 8–28.

Understanding Message Formats

Messages consist of lines of text. Graphics and binary data are not handled. No data compression is accepted.

The Message Handler (MH) package processes messages in a particular format. Although this format is different from that used by the Bell and Berkeley mail systems, the MH package can read Bell and Berkeley message files.

The general memo framework of the ARPA RFC 822 standard is used. A *message* consists of a block of information in a rigid format, followed by general text with no specified format. The rigidly formatted part of a message is the *header*, and the free-format portion is the *body*.

The header must be present, but the body is optional. Header and body are separated by an empty line; that is, two consecutive new-line characters. Within messages, the header and body can be separated by a line consisting of dashes such as:

```
To:
cc:
Subject
--------
```

The header is composed of one or more header components. Each header component can be viewed as a single logical line of ASCII characters. If the text of a header component extends across several lines, the continuation lines are indicated by leading spaces or tab characters.

The text for most formatted components (for example, the `Date:` and `Message-ID:` components) is produced automatically. The text for other components (for example, the `To:` and `cc:` components) must be entered by the user. Multiple addresses are separated by commas. A missing host or domain is assumed to be the local host or local domain.

Mail Message Components

`Date:`	The date and time of entry of the message into the transport system. The date is added by the **post** command, the **spost** command, or the mail transport system.
`From:`	The address of the author or authors. (There can be multiple authors if a `Sender:` field is present.) The address of the author is added by the **post** command or **spost** command. Replies are typically directed to addresses in the `In-Reply-To:` field or `From:` field. The former field has precedence if both fields are present.
`Sender:`	The address of the actual sender. The address of the sender is added by the **post** command, or by the **spost** command if the message already has a `From:` component. Replies are never sent to addresses in the `Sender:` field.
`To:`	Addresses of primary recipients.
`cc:`	Addresses of secondary recipients.

`Bcc:`	Addresses of secret recipients. The `Bcc:` component is not copied into the message as delivered, so these recipients are not listed. The MH package uses an encapsulation method for blind copies.
`Fcc:`	Causes the **post** command or **spost** command to copy the message into the specified folder for the sender.
`Message-Id:`	A unique message identifier added by the **post** command, or by the **spost** command if the **-msgid** flag is set.
`Subject:`	A commentary line added by the sender. The line is displayed by the **scan** command.
`In-Reply-To:`	A commentary line added by the **repl** command when the sender replies to a message.
`Resent-Date:`	The new date when a message is redistributed by the **post** command or **spost** command.
`Resent-From:`	The addresses of the new author when a message is redistributed by the **post** command or **spost** command.
`Resent-To:`	The addresses of the new recipients for a message resent by the **dist** command.
`Resent-cc:`	More recipients for a message resent by the **dist** command. (See the `cc:` and `Resent-To:` components.)
`Resent-Bcc:`	More recipients for a message resent by the **dist** command. (See the `Bcc:` and `Resent-To:` components.)
`Resent-Fcc:`	Copies the resent message into a folder. (See the `Fcc:` and `Resent-To:` components.)
`Resent-Message-Id:`	A unique identifier attached by the **post** command, or the **spost** command if the **-msgid** flag is set. (See the `Message-Id:` and `Resent-To:` components.)
`Resent:`	Annotation that the **dist** command uses when you specify the **-annotate** flag.
`Forwarded:`	Annotation that the **forw** command uses when you specify the **-annotate** flag.
`Replied:`	Annotation that the **repl** command uses when you specify the **-annotate** flag.

Each user has a mail drop that initially receives all messages. The **inc** command reads from that mail drop and incorporates the new messages found there into the user's own mail folder.

Related Information

The **dist** command, **forw** command, **inc** command, **post** command, **repl** command, **scan** command, **spost** command.

Understanding Message Handler (MH) Defaults

The Message Handler (MH) package has a multilayered default structure. When you enter an MH command, the system searches for parameters and default settings in the following sequence:

1. The system accepts all valid parameters and flags from the command line or from wherever the command was initiated.
2. The system searches the **$HOME/.mh_profile** file for the entry *command*, where *command* is the name of the program being invoked. If this profile entry exists, the system accepts all parameters that do not cancel the previously stated command line instructions.

   Note: If you invoke an MH command by specifying a link to the command, the system searches for the profile entry having that link as its name. If you invoke the command by specifying an alias, the system searches for the profile entry having the actual command, rather than the alias, as its name.
3. The system searches the **$HOME/.mh_profile** file for other entries that are applicable to the command and that have not already been overridden by previously stated parameters. If any such entries exist, the system accepts the parameters to those entries.
4. The system uses the current state information (such as the current folder and the current message) recorded in the *user_mh_directory*/**context** file for any parameters that have not been stated.
5. The system accepts systemwide defaults for any remaining unspecified parameters.

When you specify a relative path name for your MH profile, the MH programs interpret the full path name as beginning at the current directory. If you specify a relative path name for any other MH files, the MH programs interpret the full path names for those files as beginning at *user_mh_directory.*

Related Information

The **.mh_profile** file.

Understanding the Message Handler (MH) Command Output Format

Several MH commands use either a format string (similar to a **printf** command) or a format file to display their output. Each line in a format file is a format string. A format string is specified by the **–format** flag. A format file is specified by the **–form** flag. For example, the **scan** command uses a format string to organize the scan listing for each message, while the **repl** command uses a format file to organize a message reply. The default predefined files are found in the **/usr/lib/mh** directory.

An MH format string is similar to a **printf** command, but uses multiletter escape sequences beginning with the % (percent) sign.

In addition, the usual C language backslash characters (**\b**, **\f**, **\n**, **\r**, and **\t**) are recognized. To continue a format line to the next line, precede the new-line character with a \ (backslash).

The interpretation model is based on a simple machine with two registers represented by the *Number* and *String* parameters. The former contains an integer value; the latter, a string value. When an escape is processed and requires a parameter, the system reads the current value of either the *Number* or *String* parameter. Then, if the escape returns a value, the system writes either the number or the string.

Escape Sequences

Escape sequences for format files are of three types: component, function, or control. A component escape, specified as `%{`*`Name`*`}`, is created for each header found in the message being processed. For example, `%{date}` refers to the `Date:` field of the appropriate message. A component escape is always string-valued.

The control escape is one of the following:

- `%<`*`Escape`*
- `%|`
- `%>`

These control escape sequences correspond to an IF-THEN-ELSE construct. If the *Escape* variable has a nonzero value (for integer-valued escapes) or is not empty (for string-valued escapes), then everything up to `%|` or `%>` (whichever comes first) is interpreted. Otherwise, processing skips to `%|` or `%>` (whichever comes first) and starts interpreting again.

A function escape is specified as the %{*Name*} and is statically defined as follows:

Escape Defines the name of the function.

Parameter Defines what type of parameter the function expects (if any).

Returns Defines the return value type of the function (if any).

Interpretation Is a shortened explanation of what the function does.

When the *String* register is a name, the following escapes are useful:

Function Escape Sequence for the Name Field			
Escape	**Parameter**	**Returns**	**Interpretation**
nonzero	*integer*	*integer*	*Number* has a nonzero value
zero	*integer*	*integer*	*Number* is zero
eq	*integer*	*integer*	*Number* == width
ne	*integer*	*integer*	width > *Number*
gt	*integer*	*integer*	*Number* != width
null	*string*	*integer*	*String* is empty
nonnull	*string*	*integer*	*String* is not empty
putstr	*string*		Displays *String*
putstrf	*string*		Displays *String* in the specified width, for example: `%20(putstrf[Subject])`
putnum	*integer*		Displays *Number*
putnumf	*integer*		Displays *Number* in the specified width, for example: %4(putnum(Message))
msg		*integer*	Message number
cur		*integer*	Message is current
size		*integer*	Size of message
strlen	*string*	*integer*	Length of *String*
me		*string*	User's mailbox
plus		*integer*	Adds width to *Number*
minus		*integer*	Subtracts *Number* from width
charleft		*integer*	Space left in output buffer
timenow		*integer*	Seconds from 00:00:00 GMT January 1, 1970

When the *String* register is a date, the following escapes are useful:

Function Escape Sequence for the Date Field			
Escape	**Parameter**	**Returns**	**Interpretation**
sec	*string*	*integer*	Seconds of the minute
min	*string*	*integer*	Minutes of the day
hour	*string*	*integer*	Hours of the day (24-hour clock)
mday	*string*	*integer*	Day of the month
mon	*string*	*integer*	Month of the year
wday	*string*	*integer*	Day of the week (Sunday=0)
year	*string*	*integer*	Year of the century
yday	*string*	*integer*	Day of the year
dst	*string*	*integer*	Daylight savings in effect
zone	*string*	*integer*	Time zone
sday	*string*	*integer*	Day of the week known. Values are: 1 Explicit day 0 Implicit –1 Unknown
clock	*string*	*integer*	Seconds from 00:00:00 GMT January 1, 1970
rclock	*string*	*integer*	Seconds prior to current time
month	*string*	*string*	Month of the year
hmonth	*string*	*string*	Month of the year (long form)
tzone	*string*	*string*	Time zone
day	*string*	*string*	Day of the week
weekday	*string*	*string*	Day of the week (long)
tws	*string*	*string*	ARPA RFC 822 rendering of the date
pretty	*string*	*string*	A more user-friendly rendering
nodate	*string*		Could not be parsed

When the *String* register is an address, the following escapes are useful:

Function Escape Sequence for the Address Field			
Escape	**Parameter**	**Returns**	**Interpretation**
pers	*string*	*string*	Personal name of the address
mbox	*string*	*string*	Local part of the address
host	*string*	*string*	Domain part of the address
path	*string*	*string*	Route part of the address
type	*string*	*integer*	Type of host. Values are: –1 UUCP 0 Local 1 Network 2 Unknown
nohost	*string*	*integer*	No host present in the address
ingrp	*string*	*integer*	Address displayed inside a group
gname	*string*	*string*	Name of the group (present for first address only)
note	*string*	*string*	Commentary text
proper	*string*	*string*	ARPA RFC 822 rendering of the address
friendly	*string*	*string*	A more user-friendly rendering
mymbox	*string*		Address referring to user's mailbox
formataddr	*string*		Displays *String* in an address list

Example

The following format string is the default for the **scan** command:

```
%04(putnumf(msg))%<(cur)+%| %>%>{replied}-%| %>
%02(putnumf(mon{date}))%02(putnumf(mday{date}))\
%<{date} %|*>\
%<(mymbox{from})To:%14(putstr(friendly{to}))\
%|%17(putstrf(friendly{from}))%\
%{subject}<<%{body}>>
```

Note: The example has been divided into separate sections for readability. The backslashes at the end of each line indicate to MH that the string continues on the same line. (Without backslashes, each section of the string would be on a separate line.)

Following is an explanation of each section:

- `%04(putnumf(msg))%<(cur)+%| %>%>{replied}-%| %>`

  The message number is displayed in four digits. If the message is the current message, a + (plus) sign is displayed next; otherwise, a blank space is displayed. If a `Replied:` field is present, then a – (minus) sign is displayed; otherwise, a blank space is displayed.

- `%02(putnumf(mon{date}))%02(putnumf(mday{date}))`

  The month and day are displayed in two digits (zero-filled).

- %<{date} %|*>

  If no `Date:` field is present, then an * (asterisk) is displayed; otherwise, a space is displayed.

- `%<(mymbox{from})To:%14(putstr(friendly{to}))`

  If the message is from the current user, `To:` is displayed followed by a friendly rendering of the first address in the `To:` field.

- `%|%17(putstrf(friendly{from}))%`

  If the message is not from the current user, `From:` is displayed followed by the `From:` address. And finally:

- `%{subject}<<%{body}>>`

  The `subject` and initial `body` of the message are displayed.

This method of formatting messages allows you to extract individual fields and display them in the format you desire.

If you use the **–form** *File* flag when you run the **scan** command, each line in the *File* variable is treated as a format string and acts accordingly. The following files contain the **scan** command listing formats that you can look at:

- **/usr/lib/mh/scan.time**
- **/usr/lib/mh/scan.size**
- **/usr/lib/mh/scan.timely**

Related Information

The **ap** command, **dp** command, **mhl** command, **scan** command.

The **mhl.format** file.

List of Message Handler (MH) Management Commands

The **ap** command parses and reformats addresses.

The **bugfiler** command automatically stores bug reports in specified mail directories.

The **conflict** command searches for alias and password conflicts.

The **dp** command parses and reformats dates.

The **install_mh** command sets up mailbox directories.

The **post** command routes a message.

The **prompter** command invokes a prompting editor.

The **rcvdist** command sends a copy of incoming messages to additional recipients.

The **rcvpack** command saves incoming messages in a packed file.

The **rcvstore** command incorporates new mail from standard input into a folder.

The **rcvtty** command notifies the user of incoming messages.

The **sendbug** command mails a system bug report to a specified address.

The **slocal** command processes incoming mail.

The **spost** command routes a message.

These commands are described in *Commands Reference*.

List of Message Handler (MH) Files and File Formats

The **.maildelivery** file specifies actions to be taken when mail is received.

The **mhl.format** file controls the format of output for the **mhl** command.

The **.mh_profile** file customizes the MH package.

The **mtstailor** file tailors the MH environment to the local environment.

The MH Alias file format defines aliases.

These files and file formats are described in *Files Reference*

How to Customize the Message Handler (MH) Package

Prerequisite Tasks or Conditions

None.

Procedure

Each user of the Message Handler (MH) package has a user profile located in the **$HOME/.mh_profile** file. You can create personal MH default values by modifying your **$HOME/.mh_profile** file. For example, you can specify the editor to be used, the protection level for message files, or the addresses that are yours.

Related Information

The **.mh_profile** file.

How to Define How Commands Work for the Message Handler (MH) Package

Prerequisite Tasks or Conditions

None.

Procedure

You can define how MH commands work by specifying the entries in the **/usr/lib/mh/mtstailor** file. If you do not specify a value for an entry, MH uses the default value.

Note: This procedure is to be carried out by a user with root user authority.

The **/usr/lib/mh/mtstailor** file allows you to specify the following items:

- Host name of the local system
- Host name of the local system in the UUCP domain
- Location of mail drops
- Name of the file used as the mail drop
- Beginning-of-message delimiter for mail drops
- End-of-message delimiter for mail drops
- Whether support for the *MMailID* variable in the **/etc/passwd** file is enabled
- Locking discipline
- Directory for locked files
- Path name of the **sendmail** command
- Path name of the file containing the system default mail-delivery instructions
- Users to receive messages addressed to everyone.

Related Information

The **/etc/passwd** file, **mtstailor** file.

How to Define How Incoming Mail Is Processed

Prerequisite Tasks or Conditions

None.

Procedure

You can specify a list of actions to be performed on incoming mail and the conditions under which each action should be performed in the **$HOME/.maildelivery** file. For example, you can specify that messages by a particular sender be saved in a particular file, that certain messages be forwarded to particular recipients, or that undelivered messages be saved in a particular file.

Some mail delivery instructions can be specified by placing any one of the following standard Message Handler (MH) commands in the **.maildelivery** file:

rcvdist command	Forwards copies of messages to other users.
rcvpack command	Saves incoming messages in a packed file.
rcvstore command	Incorporates new mail from standard input into a folder.
rcvtty command	Notifies the user of incoming messages.

The **$HOME/.maildelivery** file is called by the **slocal** command. To activate the **slocal** command:

1. Create a file called **.forward** in your home directory.
2. Place the following line in the **$HOME/.forward** file:

   ```
   | /usr/lib/mh/slocal
   ```

   The **sendmail** command invokes the **slocal** command when it encounters this line in the **.forward** file.

If no instructions are specified in the **$HOME/.maildelivery** file or if the **$HOME/.maildelivery** file cannot be found, the **slocal** command reads the mail delivery instructions in the **/usr/lib/mh/maildelivery** file.

Related Information

The **rcvdist** command, **rcvpack** command, **rcvstore** command, **rcvtty** command, **sendmail** command, **slocal** command.

The **$HOME/.maildelivery** file.

How to Define a Message Sequence

Prerequisite Tasks or Conditions

None.

Procedure

To define a message sequence, use the **pick** and **mark** commands. The name of a sequence must contain only alphabetic characters and should not conflict with one of the Message Handler keywords. For example, you can define a sequence named `frombrenda`, but you cannot define a sequence named `new`.

You can define a maximum of ten sequences for each folder.

Related Information

The **mark** command, **pick** command.

How to Specify a Range of Messages on page 9–21, How to Use Message Sequences Defined by the Massage Handler (MH) Package on page 9–17.

How to Set an Editor for the Message Handler (MH) Package

Prerequisite Tasks or Conditions

None.

Procedure

To set an editor for the Message Handler (MH) package, specify the `Editor:` entry in the **.mh_profile** file. The editor you specify will be used by the **comp, dist, forw,** and **repl** commands. The default value is `prompter`.

Related Information

How to Customize the Message Handler (MH) Package on page 9–12.

The **comp** command, **dist** command, **forw** command, **prompter** command, **repl** command, **whatnow** command.

The **.mh_profile** file.

How to Use Message Sequences Defined by the Message Handler (MH) Package

Prerequisite Tasks or Conditions

None.

Procedure

- To specify that you want the Message Handler (MH) package to keep track of messages that you previously specified, define the entry `Previous-Sequence:` in your **.mh_profile** file. For example, place the following definition in the **$HOME/.mh_profile** file:

  ```
  Previous-Sequence: pseq
  ```

- To specify that you want the Message Handler package to keep track of messages that you incorporated into a folder but have not yet seen, define the entry `Unseen-Sequence:` in your **.mh_profile** file. For example, place the following definition in the **$HOME/.mh_profile** file:

  ```
  Unseen-Sequence: unseen
  ```

These definitions enable you to refer to the sequences *pseq* and *unseen* for each of your folders. The sequence *pseq* refers to the last message or sequence of messages that you specified. The sequence *unseen* refers to the messages for which you have run the **inc** command, but for which you have not run the **show** command.

Related Information

The **inc** command, **show** command.

The **$HOME/.mh_profile** file.

How to Define a Message Sequence on page 9–15, How to Specify a Range of Messages on page 9–21.

How to Check for Invalid Addresses

Prerequisite Tasks or Conditions

None.

Procedure

To check for invalid addresses, use the **whom** command. The **whom** command expands address headers into sets of addresses and optionally verifies whether the addresses are valid.

A valid address is an address that has a format acceptable to the mail transport system. Valid address headers do not imply that a message is deliverable.

Related Information

The **whom** command.

How to Check for Duplicate Aliases and Inappropriate Mail Drops

Prerequisite Tasks or Conditions

None.

Procedure

Use the **conflict** command to check for duplicate aliases and inappropriate mail drops. The **conflict** command is not a user command. It is designed instead to be run by other programs (such as the **mhmail** program) or by a user with root user authority.

Related Information

The **conflict** command, **sendmail** command.

How to Specify Messages

Prerequisite Tasks or Conditions

None.

Procedure

Many Message Handler (MH) commands allow you to refer to particular messages. You can specify messages in any of the following ways:

- State the numerical name of the message.

  When you store a message in a folder, the MH package provides the message with a numerical file name. You can use this numerical name when referring to that message.

- State an MH keyword that indicates a message.

  You can use the following keywords to specify messages:

first	The first message in the folder. The lowest message number is 1, but a folder does not have to contain a message by this number.
prev	The message that numerically precedes the current message.
cur	The current message.
.	The current message.
next	The message that numerically follows the current message.
last	The last message in the folder.
new	A new message. This is equivalent to the message following the `last` message in the folder. You cannot use this keyword when specifying a range of messages.
all	All of the messages in the folder.

- State the sequence name that represents one or more messages.

  Each sequence is associated with a particular folder and can be used to refer to messages in that folder only. You can specify message sequences that are designated as public sequences, message sequences that you define, and message sequences that the MH program keeps track of for you.

- Specify a range of messages.

Related Information

How to Define a Message Sequence on page 9–15, How to Specify a Range of Messages on page 9–21, How to Use Message Sequences Defined by the Message Handler (MH) Package on page 9–17.

How to Specify a Range of Messages

Prerequisite Tasks or Conditions

None.

Procedure

You can specify either an explicit range of messages (for example, from message 5 through message 11) or a general range of messages (for example, five messages beginning with message 4).

- To specify the explicit range of messages, place a – (minus) sign between the message identifier beginning the range and the message identifier ending the range.

  When you specify an explicit range, the range must contain messages.

  The following command requests a scan listing of messages `9` through `last` inclusive:

  ```
  scan 9–last
  ```

- To specify the general range of messages, place a : (colon) between the message identifier beginning or ending the range and the number of messages in the range. If the message identifier is `prev` or `last`, the message identifier ends the range. Otherwise, the message identifier begins the range. For example, the following command requests a scan listing of the last `5` messages:

  ```
  scan last:5
  ```

  The following command requests a scan listing of the first `3` messages:

  ```
  scan first:3
  ```

  Use either the + (plus) sign or the – (minus) sign to change (or explicitly state) the direction of the range.

  The + (plus) sign directly following the : (colon) indicates that the range begins with the message identifier. For example, the following command requests a scan listing of the message `prev` and the three messages following `prev`:

  ```
  scan prev:+4
  ```

  The – (minus) sign directly following the : (colon) indicates that the range ends with the message identifier. For example, the following command requests a scan listing of message `23` and the five messages preceding it:

  ```
  scan 23:–6
  ```

You cannot use the **new** keyword when specifying a range. If you specify a message that is greater than the `last` message in the folder, the Message Handler (MH) package interprets that message as the message number following `last`.

Related Information

How to Define a Message Sequence on page 9–15, How to Specify Messages on page 9–20.

How to Remove Messages and Folders

Prerequisite Tasks or Conditions

None.

Procedure

The Message Handler (MH) package provides you with two commands for removing messages and folders, respectively.

- Use the **rmm** command to remove messages.

  The **rmm** command does not actually delete messages. It renames messages by placing a , (comma) in front of the message file names. You can still use AIX commands to manipulate these files, but the comma makes the messages unavailable to the MH package. Periodically, you should delete these files. Place an entry in your **crontab** file and use the **cron** command to automatically delete all files having file names beginning with a comma.

- Use the **rmf** command to remove folders.

  The **rmf** command removes folders, but does not remove the files within the folders. Before removing a folder, make sure all files within the folder are deleted.

Since messages are AIX files and folders are AIX directories, you can also use AIX commands (such as the **rmdir**, **rm**, and **del** commands) to remove messages and folders.

Related Information

The **cron** command, **crontab** command, **rmf** command, **rmm** command.

How to Create and Use Message Drafts

Prerequisite Tasks or Conditions

None.

Procedure

When you create a message using the **comp**, **dist**, **forw**, and **repl** commands, the Message Handler (MH) package enables you to save the message in a folder until you are ready to send the message. You can specify a name for this folder using the **–draftfolder** flag.

Until you send the message, the message is considered a *message draft*. When you send the message, the MH package renames the message draft by placing a , (comma) before its file name. This renaming of the message removes the message from active draft status. (The `rmmproc:` profile entry is not consulted for this automatic renaming.)

A message draft is similar to other MH messages. You can store several message drafts in a folder, create sequences containing message drafts, and run MH commands that have a message draft as a parameter, such as the **refile**, **rmm**, and **scan** commands.

The **–draftfolder** flag does not change the current folder or the current message.

Related Information

The **comp** command, **dist** command, **forw** command, **repl** command.

How to Define a Message Sequence on page 9–15.

Chapter 10. Network Computing System (NCS)

The Network Computing System (NCS) allows you to distribute processing tasks across resources in a network or internet by maintaining databases that control the information about the resources. NCS consists of three components: the Remote Procedure Call runtime library, the Location Broker, and the Network Interface Definition Language compiler. This section provides a brief introduction to the working of NCS and its components as well as detailed information on how to configure NCS. For more detailed information on the three components of NCS, you may want to consult the chapter on NCS in *Communications Programming Concepts*.

Introduction to the Network Computing System (NCS)

The Network Computing System (NCS) is a set of tools for distributing computer processing tasks across resources in either a network or several interconnected networks (an internet). NCS is an implementation of the Network Computing Architecture, which distributes software applications across networks and internets that include a variety of computers and programming environments. Programs based on the Network Computing Architecture take advantage of computing resources throughout a network or internet, with different parts of each program executing on the computers best suited for certain tasks.

The following figure shows how the Network Computing Architecture shares processing and application data.

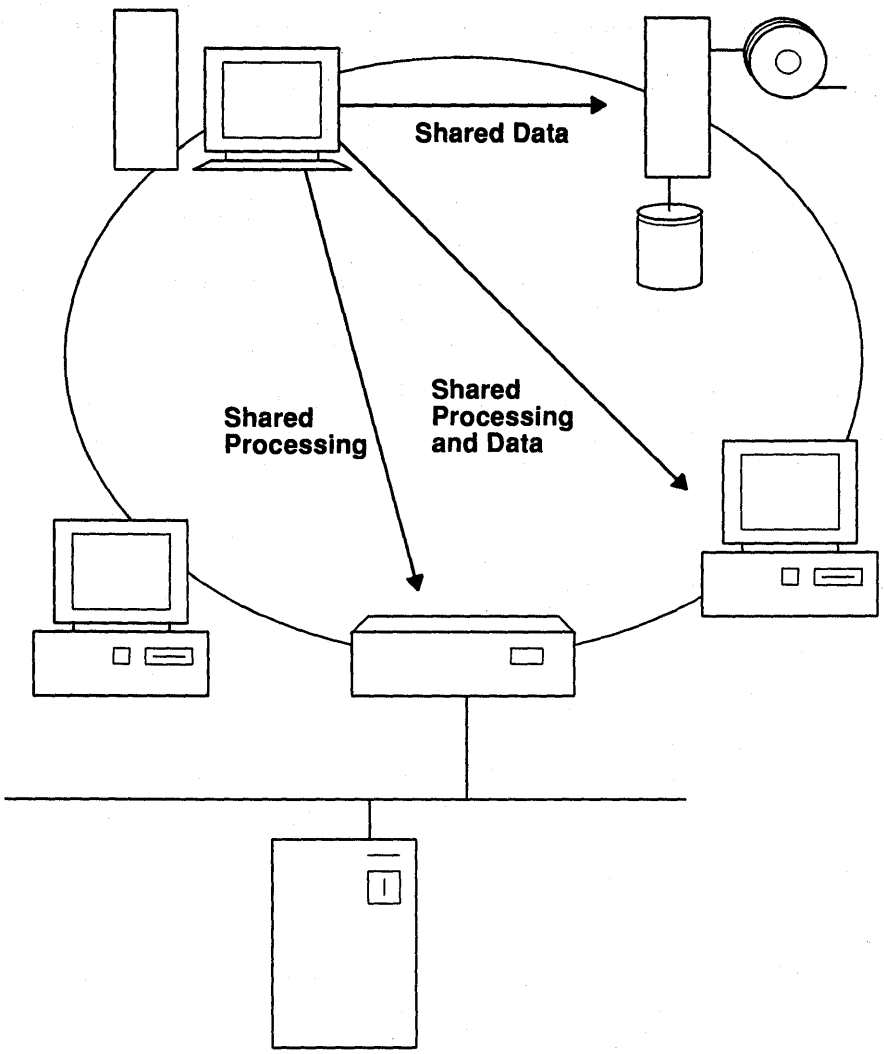


Figure 1. Distributed Computing Using NCS

Understanding NCS

NCS consists of the following components:

- Remote Procedure Call (RPC) runtime library
- Location Broker
- Network Interface Definition Language (NIDL) compiler.

The RPC runtime library and the Location Broker provide runtime support for network computing. Together these two components make up the Network Computing Kernel (NCK), which contains all the software required to run a distributed application. The NIDL compiler is a tool for developing applications.

RPC Runtime Library

The RPC runtime library provides the library routines that enable local programs to execute procedures on remote hosts. These routines transfer requests and responses between clients (the programs calling the procedures) and servers (the programs executing the procedures). When you write a distributed application, you usually do not need to use RPC routines directly. Instead, you can create an interface definition in Network Interface Definition Language and use the NIDL compiler to generate the required RPC routines.

NIDL Compiler

The NIDL compiler takes as input an interface definition written in NIDL. An interface definition specifies the interface between a user of a service and the provider of the service. It defines both the way in which a client application sees a remote service and the way in which a remote server sees requests for its service. From this definition, the NIDL compiler generates client and server stub source code and header files.

The client stub program performs the conversion between requests (and responses) that are meaningful to the client and packets that are transmitted (and received) on the network. The server stub program provides similar support for the server.

The stubs produced by the NIDL compiler contain nearly all of the remoteness in a distributed application. The stubs perform data conversions, assemble and disassemble packets, and interact with the RPC runtime library. It is much easier to write an interface definition in NIDL than it would be to write the stub code that the NIDL compiler generates from your definition.

Location Broker

The Location Broker provides information about the network or internet resources to clients. It maintains a database that contains the identities and locations of objects in the network. Through a Client Agent, the Location Broker maintains information about the local brokers that manage information about resources on the local host, the global brokers that manage information about resources available on all hosts, and the administrative tools.

Objects, Types, and Interfaces

Like the architecture on which it is based, NCS is object-oriented. An object is an entity accessed or manipulated by well-defined operations. Files, serial lines, printers, and processors can all be objects.

Programs access objects through interfaces. The programs are cast in terms of the objects they manipulate instead of the machines with which they communicate. Object-oriented programs are easy to design and can readily accommodate changes to hardware and network configurations.

- Every object has a type that specifies the class or category of the object. All objects of a type are accessed through one or more interfaces. Each interface is a set of operations that can be applied to any of the objects of that type. For example, you can classify printer queues as objects of the type printqueue, which are accessed through a **printqueue_ops** interface that includes operations to add, delete, and list jobs in the queues.

The definition of an operation specifies its input and output parameters, but not its implementation. Therefore, an operation can be implemented differently on different types of objects.

Array processors provide an example of how objects, types, and interfaces apply to NCS. You can define an **arrayproc** type. Array processor objects are accessed through either of two interfaces: a **vector_ops** interface with operations such as **vector_add** and

vector_multiply, and a **misc_ops** interface with operations such as **max_absolute_value** and **root_mean_square**.

UUIDs

NCS identifies every object, type, and interface by a Universal Unique Identifier (UUID). Each UUID is a 16-byte quantity identifying the host on which the UUID is created and the time at which it is created. Six bytes identify the time, two are reserved, and eight identify the host.

The Network Computing Kernel (NCK) includes a **uuid_gen** utility that generates UUIDs as ASCII strings or as data structures defined in the C or Pascal programming language. The string representation used by the NIDL compiler and by NCK utilities consists of 28 hexadecimal digits arranged as in this example:

```
3a2f883c4000.0d.00.00.fb.40.00.00.00
```

Clients and Servers

A client is a program that makes remote procedure calls. A remote procedure call requests that a particular operation be performed on a particular object. A server is a program that implements interfaces. The server process listens for requests for each interface's operations. When it receives a request from a client, the server executes the procedures that perform the operation and sends a response to the client.

Network Communications

The communications between systems in an NCS environment are handled through the RPC runtime library. It is possible that one program can access different hosts that listen on two different ports or have two different addresses.

In the NCS environment, RPC uses sockets for interprocess communications. A socket is an end point for communications, in the form of a message queue. An RPC server listens on one or more sockets. It receives any message sent to a socket on which it is listening. Messages can be broadcast to sockets at several hosts on the local network. Broadcasting is often used when the location of an object is not known.

The following figure illustrates RPC communications using sockets. It shows two servers running on one host and several clients on other hosts. Server 1 listens on two sockets: one socket uses the Internet Protocol (IP); the other uses another network communications interface. Server 2 listens on a socket that uses IP.

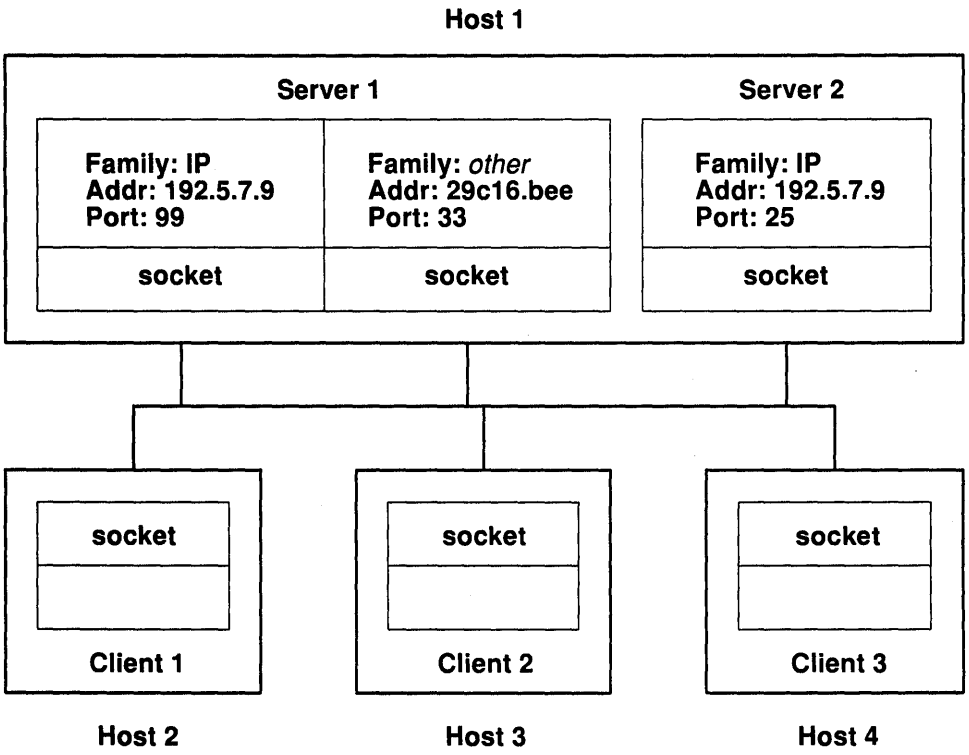


Figure 2. RPC Communications Using Sockets

Each socket is identified uniquely by a socket address. A socket address, sometimes called a sockaddr, is a data structure that specifies the following information about a socket:

- Address family, also called the protocol family, which determines the communications protocol used to deliver messages and the structure of the addresses used to represent communications end points.
- Network address, which is a value that, given the communications protocol, uniquely identifies a host on one or more interconnected networks.
- Port number, which specifies a communications end point within the host. The terms port and socket are synonymous, but port number and socket address are not. A port number is one of the three parts in a socket address. For example, a port number can be represented as the character string `77`, while a socket address can be represented as `ip:myhost[77]`.

The following figure illustrates the socket address structure for a domain socket address and an IP socket address.

Domain Socket Address

Family	Port	Network Address	
16-bit integer	16-bit integer	Network	Host
		32-bit integer	32-bit integer

IP Socket Address

Family	Port	Network Address
16-bit integer	16-bit integer	32-bit integer

Figure 3. Domain Socket Address and IP Socket Address Structures

Well-Known and Opaque Ports

Interfaces can be designed and implemented with a particular port number *built in.* The port used in such an interface is called a well-known port. Clients of the interface always send to that port, and servers always listen on that port. Some well-known ports are assigned to particular servers by the administrators of a communications protocol. For example, the administrators of the Internet Protocols have assigned port number 23 to the server for the telnet remote login facility. All telnet servers listen on this well-known port, and all telnet user programs send to it.

Well-known ports are an effective way to coordinate communication between clients and servers if portability to other networks and coexistence with other services are of little concern. However, the number of ports in each protocol family is limited. Unless the assignment is obtained from a central administrator, an application's well-known port number is liable to conflict with that of another program. The NCS Location Broker circumvents this problem by allowing you to locate services easily without direct use of well-known ports. It uses one well-known port to listen for requests. Clients and servers can locate a broker by broadcasting to this port.

NCS enables a server to use ports that the RPC runtime software assigns dynamically. After a server registers this assignment with the Location Broker, a client can then obtain the server's socket address from the broker. Since there is no need for either the client or the server to know a port number, the number is said to be opaque.

The RPC Paradigm

Remote procedure calls extend the procedure call mechanism from a single system to a distributed computing environment. The calls distribute the execution of a program among multiple computers in a way that is transparent to the application-level code.

The following figure shows the flow of ordinary local procedure calls between the calling client and the called procedures.

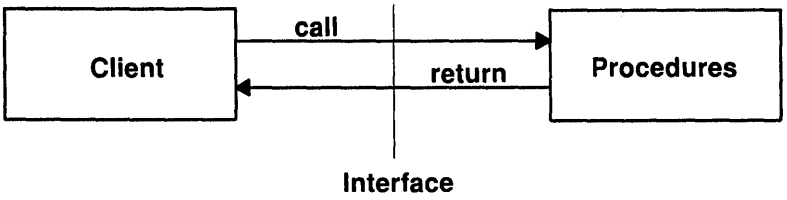


Figure 4. Single-Process Procedure Call Flow

The following figure shows the same flow for remote procedure calls and illustrates how RPC hides the remote aspects of a call from the calling client. The client application requests a procedure by using standard calling conventions, as if the procedure were a part of the local program, the procedure is, however, executed by a remote server. The client stub acts as the local representative of the procedure, organizing the data into a format that can be transmitted to the server and using RPC runtime library routines to communicate with the server. Similar activities occur within the server process.

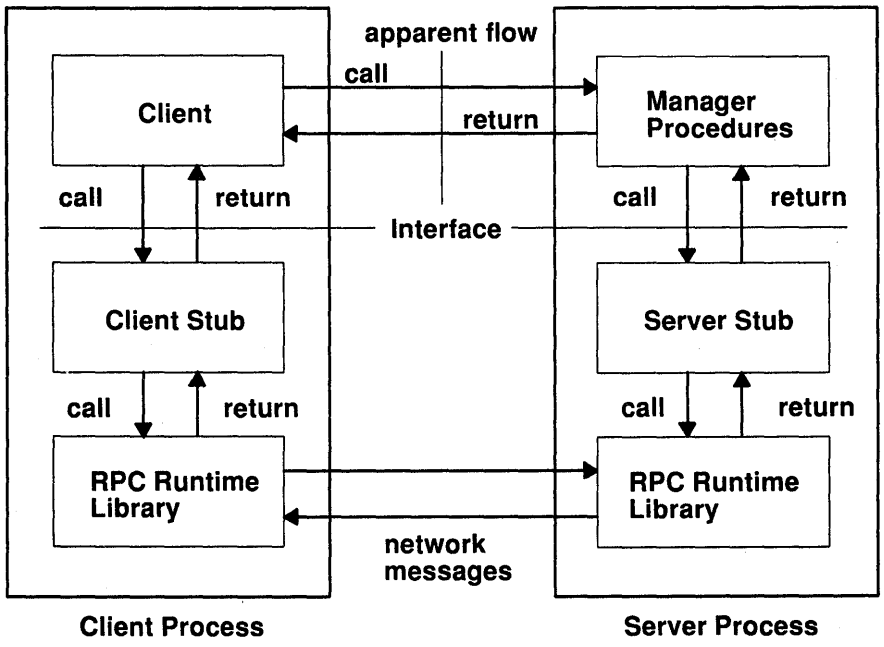


Figure 5. Remote Procedure Call Flow

Interfaces

An interface consists of procedure names and signatures. It defines the calling syntax that is used both by the client and by the remote procedures. An interface is independent of the mechanism that conveys the request between the client and the procedures; it is also independent of the way in which the procedures perform the operations. The server that implements an interface's operations is said to export the interface. The program that requests the operations imports the interface.

For example, suppose that **print** is a print queue manipulation interface used to manipulate queues for several types of printers. One printer type is **laser**, and objects of this type are **my_laser**, **your_laser**, and **public_laser**. The **print** interface includes the **print$add_to_queue**, **print$delete_from_queue**, and **print$check_queue** operations.

A remote matrix arithmetic package is another example of an interface. An array processor exports a set of matrix operations as an interface. The array processor is the object, and its type is **arrayproc**. Array processor objects are accessed through a **vector_ops** interface with operations such as **vector_add** and **vector_multiply**. The **arrayproc** type might have other interfaces, for example, a **misc_ops** interface, with operations such as **max_absolute_value** and **root_mean_square**, and a **scalar_ops** interface for scalar arithmetic. Client programs on various hosts import the **vector_ops** interface by making calls such as **vector_add**. The programs run on the local hosts, but all matrix operations run on the remote array processor.

Clients, Servers, and Managers

An RPC client is a program that makes remote procedure calls to request operations. A client does not know how an interface is implemented. The client is not required to know the location of the server exporting the interface either.

An RPC server is a process that implements the operations in one or more interfaces. The server is the module to which the RPC runtime library sends an operation request packet, and from which this library receives a response containing the results of the operation.

A server can export a single interface or multiple interfaces as explained in the array processor example discussed previously in Interfaces on page 10–7. The following figure illustrates a server that exports two interfaces. A server can export an interface for a single object or for multiple objects. In the array processor example, there is only one object, the array processor. A file server, however, manages many file objects.

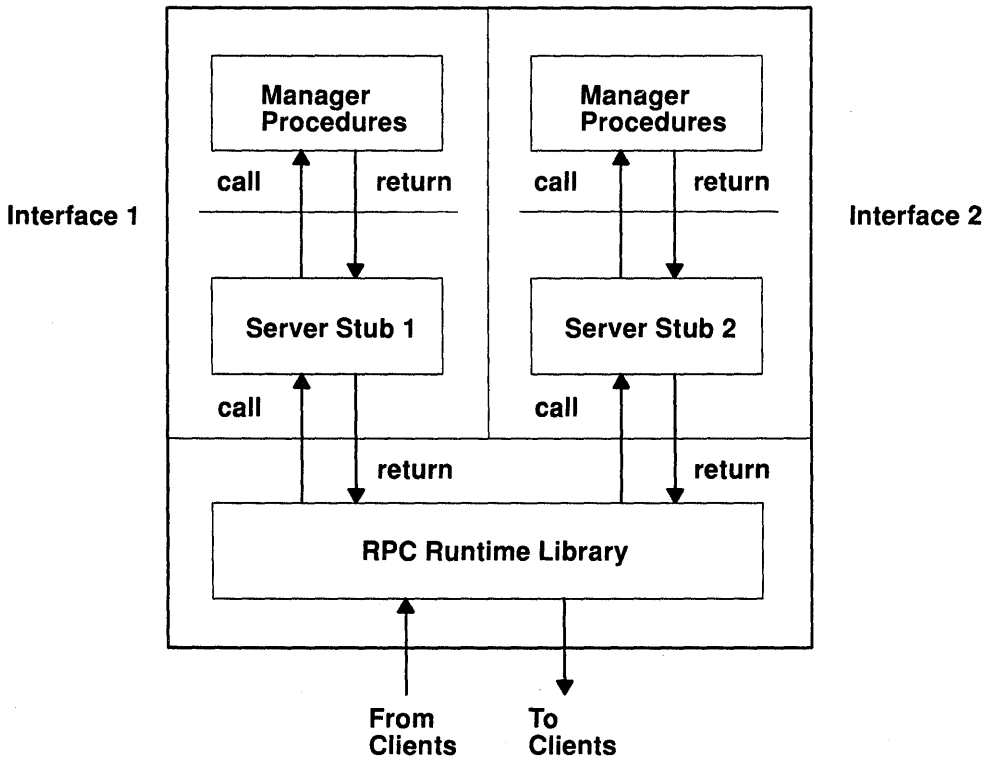


Figure 6. An RPC Server Exporting Two Interfaces

A server can also be a client. It can even be a client of itself. The client and server play symmetrical roles in the RPC paradigm. However, their program structures are asymmetrical.

The client consists of two parts:

- The application code (labeled client), which makes calls to be executed remotely
- The stub code (labeled client stub), which uses the RPC runtime library to have these calls executed.

The server has the following parts:

- The manager code, which corresponds to the client application code
- A stub, which corresponds to the client stub
- The code that initializes the server process itself.

Reference to the server means the whole server process. However, manager code refers directly to the procedures that actually implement the server operations.

Identifying Objects and Servers

When a client makes a remote procedure call to request that a particular operation be performed on a particular object, the following information is required to transmit the call from the RPC runtime library:

- The object on which the operation is to be performed
- The server that exports the interface containing the operation.

This information about the object and the server is represented in the client process by a handle. Handles are created and managed by RPC library routines. Once a handle is created, it always represents the same object. The handle can, however, represent different servers at different times, although it is not required to specify a server at all. The representation of the server in a handle is called the binding. To bind a handle is to establish its representation of the server.

RPC Handles

An RPC handle is a pointer to an opaque data structure that includes the information required to access a remote object. Clients and servers do not manipulate this structure directly, but through RPC runtime library calls. The following figure shows an RPC handle.

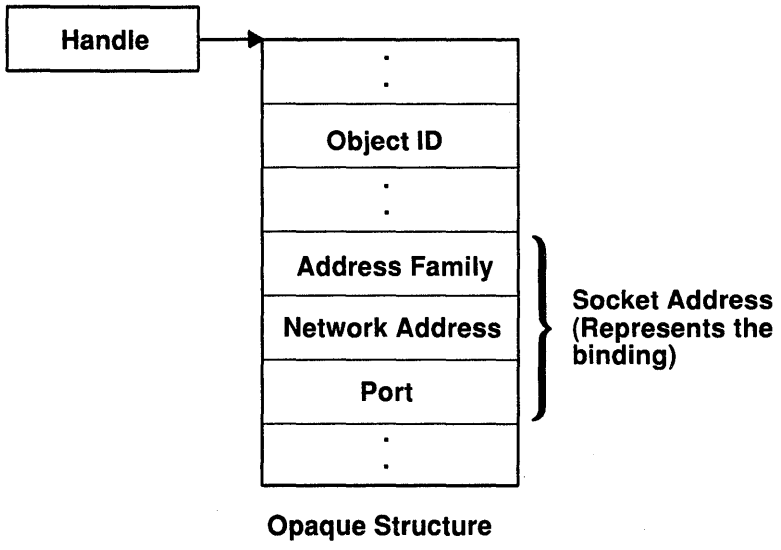


Figure 7. RPC Handle

The RPC Binding

A binding, and therefore the RPC handle that includes the binding, can exist in three states:

- Unbound (or allocated)
- Bound-to-host
- Bound-to-server (or fully bound).

An unbound or allocated handle identifies an object but does not identify its location. When a client uses an unbound handle to make a remote procedure call, the runtime library broadcasts a message to all hosts on the local network. Any host that supports the requested interface to the object (such as the interface that contains the called operation) can respond. The client runtime library accepts the first response that it receives.

A bound-to-host handle identifies the object and the host but does not represent the specific server that exports the interface to the object. When a client uses a bound-to-host handle to make a remote procedure call, the runtime library sends a message to the forwarding port on the specified host. If a server that exports the required interface to the object is registered with the host's Local Location Broker, the message is forwarded to the required server.

A bound-to-server or fully bound handle identifies the object and the server. When a client uses a fully bound handle to make a remote procedure call, the runtime library sends the message directly to the socket address identified by the handle.

In all cases, whenever the client RPC runtime library receives a response from a server, it binds the handle to the server socket address. Therefore, RPC handles are fully bound whenever a remote procedure call returns, and the client does not need to use the broadcasting or forwarding mechanism for subsequent calls to the server.

The following table shows, for each possible binding state of a handle when a remote procedure call is made, the information that the handle represents, the delivery mechanism of the remote procedure call, and the binding state when the procedure call returns.

Handles and Binding States

Binding State on Call	Information	Delivery	Binding State on Return
Allocated	Object	Broadcast to all hosts	Fully bound
Bound-to-host	Object Host	Sent to host's Location Broker forwarding port	Fully bound
Fully bound	Object Host Server	Sent to specific server port	Fully bound

Stubs

Both clients and servers are linked (in the sense of combining object modules to form executable files) with stubs. Stubs enable the clients and servers to use the RPC facilities as transparently as possible, which makes remote invocations look almost local. The client stub stands in for the remote procedures in the client process. The server stub stands in for the client in the server process. This means that, when a client makes a remote procedure call, it actually calls a routine in the client stub. The client stub calls an RPC runtime routine to send the request to the server. Similarly, the server RPC runtime library calls the server stub when it receives an RPC packet, and the server stub then calls manager code that executes the requested procedure.

The stub program modules transfer remote procedure calls and responses between an RPC client and the manager procedures that implement an interface. The modules convert data between the procedure call format specified by the interface definition and the format required by the RPC runtime routines. The modules also issue the RPC runtime library calls required for communication between the client and the server.

When a client calls an interface operation, such as **vector_add** from the array processor example, it actually calls a routine in the client stub. The client stub does the following:

1. Establishes the binding between the client and the server if the client has not explicitly created a binding.
2. Marshalls, or copies into an RPC packet, the input parameter values.
3. Calls an RPC runtime procedure to send the packet to the server stub and await a reply.
4. Receives the reply packet.
5. Unmarshalls the output parameter values into the format expected by the client. The format is specified in the interface definition.
6. Converts the output's data representation into a form that is meaningful to the client if the server uses a different format (for example, converts characters from EBCDIC to ASCII).
7. Returns to the client.

Similarly, the RPC runtime library calls a server stub routine when the server receives a request from the client. The server stub then does the following:

1. Unmarshalls the input parameter values into the format expected by the server. The format is specified in the interface definition.
2. Converts the input's data representation into a form that is meaningful to the server if the client uses a different format (for example, converts characters from ASCII to EBCDIC).
3. Calls the manager procedure that implements the operation.
4. Marshalls the output parameter values into an RPC packet.
5. Returns the packet to the RPC runtime library for transmission to the client stub.

The following figure illustrates these operations in the context of the figure illustrating a remote procedure call flow and the figure illustrating a server exporting two interfaces.

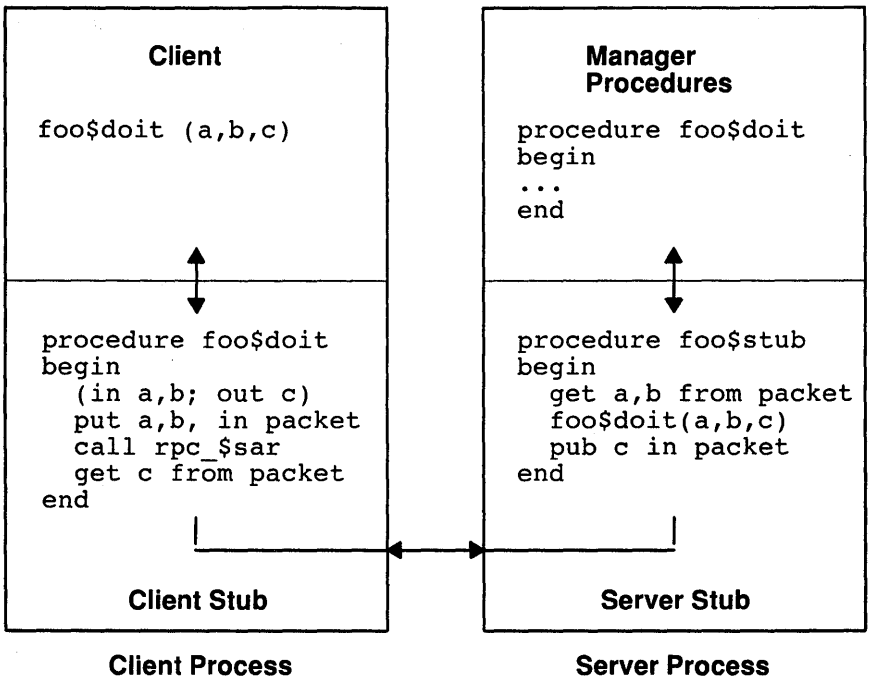


Figure 8. Client and Server Stub Operations

NCS provides a compiler that automatically generates source code for both the client and the server stubs from a definition of the interface written in Network Interface Definition Language (NIDL). Interface Definitions and the NIDL Compiler in *Communications Programming Concepts* provides more detailed information about the NIDL compiler and the stubs that it generates.

NCS Daemons and Utilities

The description of each NCS daemon and utility program includes the following information:

- A **Purpose** section identifying the use of the daemon or utility
- A **Syntax** section showing the syntax or location of the program
- A **Description** section giving an explanation of the usage
- A **Files** section describing any files related to the operation of the program
- A **Related Information** section identifying other related concepts in this documentation.

Some of the daemons and utilities also include descriptions of subcommands that are used along with the program, and examples of command usage.

List of Daemons and Utilities

- **lb_admin** command
- **llbd** daemon
- **nidl** command
- **nrglbd** daemon
- **uuid_gen** command.

These commands and daemons are described in *Commands Reference.*

Configuring NCS

This section describes how to configure the NCS program, where the files and directories are stored, and how to configure the Location Broker on your network or internet.

Notes:

1. To install NCS, see How to Install AIX for RISC/6000 Licensed Program Optional Program Products in *General Concepts and Procedures*.
2. After NCS installation is complete, all binary programs are in the **/usr/lpp/ncs/bin** directory. There is a symbolic link from the **/usr/lpp/ncs/bin** directory to the **/etc/ncs** directory for easier access to programs. All sample programs are installed in the **/usr/lpp/ncs/examples** directory.

How to Configure NCS

To configure NCS, complete the following steps:

1. Add the **/etc/ncs** directory to the **PATH** variable.
2. Working in AIX Version 3.1, start the TCP/IP interface.
3. Start the NCS daemons from the **/etc/rc.ncs** file, as follows:
   a. Start the Local Location Broker (LLB) by uncommenting the following line:

      ```
      # startsrc -s llbd
      ```

   b. Start the Global Location Broker (GLB) by uncommenting the following line:

      ```
      # startsrc -s nrglbd
      ```

   OR

   Start both daemons at once by uncommenting the following line:

   ```
   # startsrc -g ncs
   ```

   Notes:

   1. To configure more than one Global Location Broker on a network, see How to Configure the Location Broker on page 10–15.
   2. Refer to the following section for information on the **startsrc** command.

The System Resource Controller (SRC)

The System Resource Controller (see *General Concepts and Procedures*) operational control component provides a means of specifying the desired software environment of a system by controlling how subsystems are started and by notifying related subsystems of a failing subsystem. SRC also supports a consistent set of commands to start, stop, and query the status of subsystems.

Once SRC is active, start and stop commands may be used to start the NCS daemons and control their operation. NCS supports three basic SRC commands:

- The **startsrc** command starts the execution of NCS daemons.
- The **stopsrc** command stops the execution of NCS daemons.
- The **lssrc** command displays status.

SRC uses the **SIGTERM**, **SIGUSR1**, and **SIGUSR2** signals to communicate with the NCS daemons. These signals correspond to the three variations of the **stopsrc** command.

How to Configure the Location Broker

When planning a Location Broker configuration, consider the following:

- Where Location Broker daemons must run:
  - A GLB daemon must run on at least one host in any network where NCS-based programs run.
  - An LLB daemon must run on any host where a GLB or any other NCS-based server runs.

    Note: It is, however, recommended to run an LLB daemon on every host where NCS-based programs run.
- GLB configuration:
  - An **nrglbd** daemon communicates with its clients using the IP protocol.
  - Normally, only one **nrglbd** daemon should run on a network. On a large network, however, multiple **nrglbd** daemons may be desirable to improve accessibility. Each host on the network should then have a file named **/etc/ncs/glb_sites** that specifies the **nrglbd** daemon to be accessed by that host. This file should contain lines of the format:

    ```
    ip:hostname
    ```

    where each line indicates a host to check for an active **nrglbd** daemon. The entries in the **glb_sites** file are processed sequentially until an active daemon is found or the end of the file is reached.

    In an internet of foreign systems, one **nrglbd** daemon can run on each network, but each **nrglbd** daemon can service only the hosts in its own network.
  - If an AIX node is acting as a gateway between two networks, an **nrglbd** daemon running on that node can serve clients on both networks.

Note: For more information on the Location Broker, see The Location Broker in *Communications Programming Concepts*.

Network Configuration

The following figure illustrates a network that contains both AIX and foreign systems. Here, GLB service is provided by a **nrglbd** daemon running on the AIX host AIX1. An **llbd** daemon runs on this host and on every other host that runs NCS-based programs.

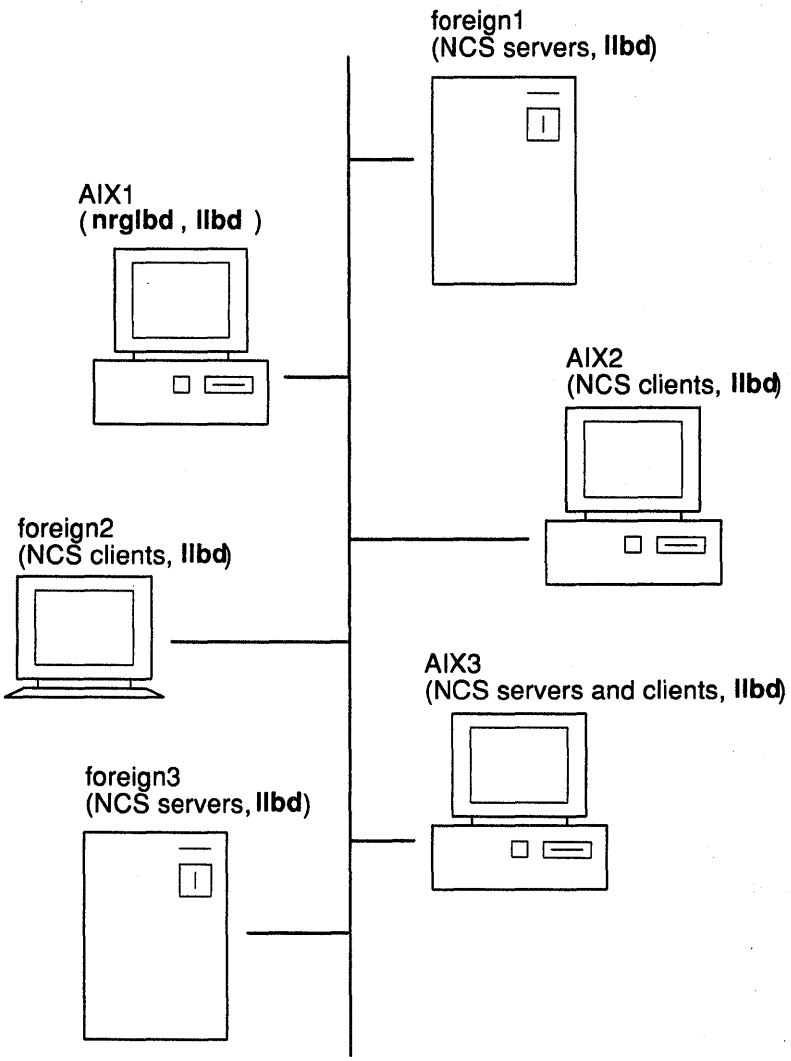


Figure 9. A Network of AIX Systems and Foreign Systems

Internet Configuration

The following figure illustrates an internet that consists of two networks. One network contains only AIX systems. The other contains both AIX and foreign systems. The AIX host AIX2 acts as a gateway between the two networks. The **nrglbd** daemon running on AIX2 provides GLB service to both networks. An **llbd** daemon runs on this host and on every other host that runs NCS-based programs.

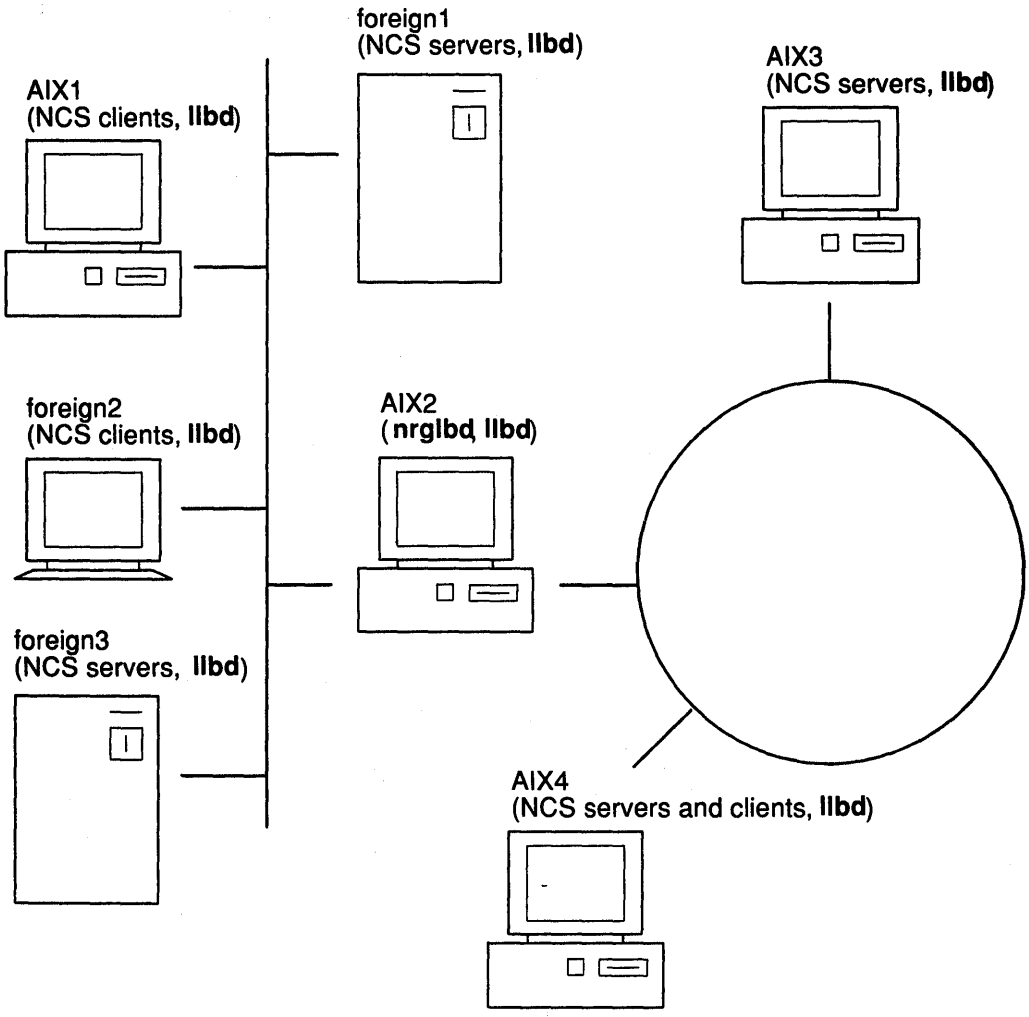


Figure 10. An Internet Containing AIX and Foreign Systems

Running the Local Location Broker Daemon

To allow remote access to the LLB database or to support the Location Broker forwarding function, the **llbd** daemon must be started on each host.

The **llbd** daemon can be started in one of two ways:

- By a person with root user authority entering on the command line:

  ```
  /etc/ncs/llbd &
  ```

- Through the System Resource Controller (SRC), by entering on the command line:

  ```
  startsrc -s llbd
  ```

To start the LLB automatically each time the node reboots, uncomment the following line in the node's **/etc/rc.ncs** file:

```
# startsrc -s llbd
```

Note: To start the **llbd** daemon during NCS configuration, see How to Configure NCS on page 10–14.

Running the Global Location Broker Daemon

An **nrglbd** daemon must run on at least one host in a network. In an internet, at least one GLB database must run in each network that supports NCS. The LLB daemon (**llbd**) must be running on the local node when the **nrglbd** daemon is started.

The **nrglbd** daemon can be started in one of two ways:

- By a person with root user authority entering on the command line:

  ```
  /etc/ncs/nrglbd &
  ```

- Through the System Resource Controller (SRC), by entering on the command line:

  ```
  startsrc -s nrglbd
  ```

To start the GLB automatically each time the node reboots, uncomment the following line in the node's **/etc/rc.ncs** file:

```
# startsrc -s nrglbd
```

Note: To start the **nrglbd** daemon during NCS configuration, see How to Configure NCS on page 10–14.

Glossary

address family. A set of communications protocols that use a common addressing mechanism to identify end points. This term is often used synonymously with *protocol family.*

allocate (a handle). To create a Remote Procedure Call (RPC) handle that identifies an object.

bind. To set a binding. NCS provides two library routines that bind: **rpc_$bind**, which both creates and binds a handle, and **rpc_$set_binding**, which requires a handle as an input parameter.

binding. A temporary association between a client and both an object and a server that exports an interface to the object. A binding is meaningful only to the program that sets it and is represented by a bound handle.

broker. A server that manages information about objects and interfaces to the objects. A program that wishes to become the client of an interface can use a broker to obtain information about servers that export the interface. Location brokers are brokers.

client. A user of an interface. In the context of this manual, a program that makes remote procedure calls.

Client Agent. See *Location Broker Client Agent.*

entry point vector (EPV). A record whose fields are pointers to procedures that implement the operations defined by an interface.

EPV. See *entry point vector.*

export. To provide the operations defined by an interface. A server exports an interface to a client. See also *import.*

GLB. See *Global Location Broker.*

Global Location Broker (GLB). Part of the NCS Location Broker. A server that maintains global information about objects on a network or an internet.

handle. A data structure that is a temporary local identifier for an object. You create a handle by allocating it. You make a handle identify an object at a specific location by binding it.

host. A computer that is attached to a network.

host ID. An identifier for a host. A host ID uniquely identifies a host within an address family on a network, but does not identify the network. A host ID is not necessarily sufficient to establish communications with a host. See also *network address.*

idempotent. A class of operations. An operation is idempotent if its results do not affect the results of any operation. For example, a call that returns the time is idempotent.

import. To request the operations defined by an interface. A client imports an interface from a server. See also *export.*

interface. A set of operations. The Network Computing Architecture specifies a Network Interface Definition Language for defining interfaces.

Internet Protocol (IP). The protocol that provides the interface from the higher level host-to-host protocols to the local network protocols. Addressing at this level is usually from host to host.

IP. See *Internet Protocol.*

LLB. See *Local Location Broker.*

Local Location Broker (LLB). Part of the NCS Location Broker. A server that maintains information about objects on the local host. The LLB also provides the Location Broker forwarding facility.

Location Broker. A set of software including the Local Location Broker, the Global Location Broker, and the Location Broker Client Agent. The Location Broker maintains information about the locations of objects.

Location Broker Client Agent. Part of the NCS Location Broker. Programs communicate with Global Location Brokers and with remote Local Location Brokers using the Location Broker Client Agent.

marshall. To copy data into a Remote Procedure Call (RPC) packet. Stubs perform marshalling. See also *unmarshall.*

NCK. See *Network Computing Kernel.*

NCS. See *Network Computing System.*

network address. A unique identifier (within an address family) for a specific host on a network or an internet. The network address is sufficient to identify a host, but does not identify a communications end point within the host.

Network Computing Architecture. A set of protocols and architectures that support distributed computing.

Network Computing Kernel (NCK). The combination of the RPC runtime library and the Location Broker, which contain the necessary pieces required to run distributed applications.

Network Computing System (NCS). A set of software tools developed by Apollo Computer Inc. that conform to the Network Computing Architecture. These tools include the Remote Procedure Call runtime library, the Location Broker, and the NIDL compiler.

Network Interface Definition Language (NIDL). A declarative language for the definition of interfaces. A component of the Network Computing Architecture. NIDL has two forms, a Pascal-like syntax and a C-like syntax.

NIDL. See *Network Interface Definition Language.*

NIDL compiler. An NCS tool that converts an interface definition, written in NIDL, into several program modules, including source code for client and server stubs. The NIDL compiler accepts interface definitions written in either syntax of NIDL. It generates C source code and C or Pascal header files.

object. An entity that is manipulated by well-defined operations. Disk files, printers, and array processors are examples of objects. Objects are accessed though interfaces. Every object has a type.

object UUID. A UUID that identifies a particular object. Both the RPC runtime library and the Location Broker use object UUIDs to identify objects.

operation. A procedure through which an object is accessed or manipulated. An operation is defined syntactically by its name and its parameters but not by its implementation.

PFM. See *program fault management.*

port. A specific communications end point within a host. A port is identified by a port number. See also *socket.*

program fault management (PFM). A subsystem of NCS that allows a user to set up cleanup routines when an application fails to complete successfully.

protocol family. A set of related communications protocols, for example, the Department of Defense Internet Protocols. All members of a protocol family use a common addressing mechanism to identify end points. This term is often used synonymously with *address family.*

register (an interface). To make an interface known to the RPC runtime library, and thereby available to clients through the RPC mechanism. The **rpc_$register** routine registers an interface.

register (an object). To enter an object and its location in the Location Broker database. The **lb_$register** routine registers an object with the Location Broker. A program can use Location Broker lookup routines to determine the location of a registered object.

remote procedure call. An invocation of a remote operation. You can make remote procedure calls between processes on different hosts or on the same host.

Remote Procedure Call runtime library. The set of **rpc_$** library routines that NCS provides to implement a remote procedure call mechanism.

RPC. See *remote procedure call.*

server. A process that implements interfaces. In the context of this manual, a server whose procedures can be invoked from remote hosts. A server exports one or more interfaces to one or more objects.

set (a binding). To associate an allocated Remote Procedure Call (RPC) handle with a specific socket address.

socket. A port on a specific host; a communications end point that is accessible through a protocol family's addressing mechanism. A socket is identified by a socket address.

socket address. A data structure that uniquely identifies a specific communications end point. A socket address consists of a port number and a network address. It also specifies the address family (protocol family).

stub. A program module that transfers remote procedure calls and responses between a client and a server. Stubs perform marshalling, unmarshalling, and data format conversion. Both clients and servers have stubs. The NIDL compiler generates client and server stub code from an interface definition.

type. A class of object. All objects of a specific type can be accessed though the same interface or interfaces.

type UUID. A UUID that permanently identifies a particular type. Both the RPC runtime library and the Location Broker use type UUIDs to specify types.

UID. See *user number.*

Universal Unique Identifier (UUID). A 128-bit value used for identification. NCS uses UUIDs to identify interfaces, objects, and types.

unmarshall. To copy data from an RPC packet. Stubs perform unmarshalling. See also *marshall.*

user number. A number that uniquely identifies a user to a system.

UUID. See *Universal Unique Identifier.*

Chapter 11. Network File System

The Network File System (NFS) is a distributed file system that allows users to access files and directories located on remote computers and treat those files and directories as if they were local. This chapater contains information to help you manage NFS and NIS. NIS is a network information service that provides NFS with the information necessary for file sharing. The information in this chapter includes coneptual explanations of NFS/NIS and procedural information to help you set up and configure this network application.

Network File System (NFS) Overview for System Management

NFS is independent of machine types, operating systems, and network architectures. NFS achieves this independence through the use of *remote procedure calls* (RPC) defined through the RPC protocol. RPC is built on top of the External Data Representation (XDR) protocol, which standardizes the various data types used in remote communications. NFS uses Transmission Control Protocol/Internet Protocol (TCP/IP) to implement data transmissions, so AIX TCP/IP must be installed on your system before NFS can work.

The NFS component consists of commands and daemons. The NFS commands obtain information about, and control remote file systems. The NFS daemons handle the remote file system mounts and remote file operations.

NFS works as a client-server relationship. The computers that make their files and resources available for remote access are called *servers*. The process that makes the files and resources available is called *exporting*. The computers, or the processes they run, that use the servers' files are considered *clients*. Clients mount and use the files that a server exports. A computer can be both a server and a client.

NFS Servers and Exporting

NFS servers export file systems to NFS clients. Exporting a directory makes the directory available for use by other machines (clients) on the network. Access to exported directories can be restricted to specified clients.

A server's **/etc/exports** file lists these available directories, the clients allowed to access them, and any access restrictions. When you start an NFS server, the **/etc/rc.nfs** script automatically starts the **exportfs** command. This command then looks at the **/etc/exports** file and informs the server's kernel about the permissions applicable to each exported directory.

You can also export and unexport a directory after startup by using the **exportfs** command on the server or by using the **smit** command.

Understanding the Stateless Server

The NFS server is *stateless*. The server does not have to record transaction information about its clients, the transactions completed, or the files operated on. Because servers do not have to keep track of the state of each file, there is no **open** operation in NFS itself. NFS relies on the client to remember the information for later NFS uses.

An example of a stateless server operation is the **mount** command. A client can use the **mount** command to build a view of a file system located on its local devices. The client can use the **mount** command with NFS to build a view of a remote file system. This operation is

stateless because the command gives all the information needed for the client to establish contact with a server. It does not require the server to keep any information.

An NFS server can also be a client of another server. Servers can only answer requests about their own exported file systems. They cannot serve as messengers between a client and another server.

Users can use AIX commands to create, remove, read, write, and set file attributes for remote files and directories.

Clients and Mounting

Clients access files on the server by *mounting* the server's exported directories. When a client mounts a directory, it does not make a copy of that directory. Rather, the mounting process uses a series of remote procedure calls to enable a client to transparently access the directories on the server's disk.

Binding is the process by which a client locates a server that exports the information it wants, and then sets up communication between itself and that server. NFS binding occurs during an NFS mount.

A client can mount a directory when it boots or explicitly mount a directory when a user issues a **mount** command. Specified entries in the **/etc/filesystems** file allow client mounts at startup time. You can also explicitly mount a file system during a work session by using the **mount** and **umount** commands.

Understanding NFS Mounting

The following list summarizes the activities that take place when an NFS client mounts a directory. On the server, NFS service is controlled by the **exportfs** command and the **rpc.mountd** and **nfsd** daemons. On a client, NFS service is handled by the **mount** command and the **biod** daemons.

1. When the server restarts, the **rc.nfs** script executes the **exportfs** command, which reads the server's **/etc/exports** file, and then tells the kernel which directories are to be exported and which access restrictions they require.
2. The **rpc.mountd** daemon and several **nfsd** daemons are started automatically by the **rc.nfs** script when the server boots.
3. When the client restarts, its **/etc/filesystems** file is automatically read by the **mount** program.
4. The **mount** program then requests that the server allow the client to access the directories in the client's **/etc/filesystems** file.
5. The server's **rpc.mountd** daemon handles the client's mount requests.
6. If the requested directory is available to that client, the **rpc.mountd** daemon sends the client's kernel an identifier called a *file handle*.
7. When the client does a file operation, it sends the file handle to the server, where it is read by one of the **nfsd** daemons to process the file request.
8. The **nfsd** daemons know how a directory is exported from the information sent to the server's kernel by the **exportfs** command. These daemons allow the client to access the directory, according to its permissions, by way of the file handle.

Understanding Network Services Included with NFS

The NFS software package includes commands and daemons for NFS, NIS, and other services. These services are built with the XDR and RPC protocols. These services use a

daemon called the **portmap** daemon. These items form a basis for the network packages included with NFS and the development of new network services.

XDR (eXternal Data Representation) is a specification for the standard transmission of data. This usually applies to network transmission. Together with the RPC protocol, it provides a standard I/O library for interprocess communication.

The RPC or Remote Procedure Call protocol is a library of procedures that allows one process (the client process) to direct another process (the server process) to execute a procedure call as if the client process had executed the procedure call in its own address space. Since the client and the server are two separate processes, they no longer have to exist on the same physical system.

With these two protocols, other protocols or applications can be defined. An example of this is the Network File System. NFS is defined as a set of RPC calls. The server is defined such that it can service certain types of calls and the client makes these calls based on the file system operations that are done by the client process. NFS is just another RPC application.

The IBM NFS network information service (NIS) is another example of an RPC based application. The NIS server replies to RPC calls that are made to retrieve information from its databases. The client NIS process can be any process that requests system information.

The portmap daemon

Each RPC application has associated with it a program number and a version number. These numbers are used to communicate with a server application on a system. The client, when making a request from a server, needs to know what port number that server is accepting requests on. This port number is associated with the UDP or TCP protocol that is being used by the service. The client knows the program number, the version number, and the system name or host name where the service resides. The client needs a way to map the program number, version number pair to the port number of the server application. This is done with the help of the **portmap** daemon.

The **portmap** daemon runs on the system where the service application is executing. When the service starts its execution it registers with the **portmap** daemon. As a function of this registration the service supplies the program number, version number, and UDP or TCP port number of its application. The **portmap** daemon keeps a table of server applications. When the client tries to make a request of the server, it first contacts the **portmap** daemon to see where the service is residing. The **portmap** daemon listens on a well-known port so the client does not have to worry about finding the **portmap** daemon. The **portmap** daemon responds to the client with the port of the service that the client is requesting. The client upon receipt of the port number is able to make all of its future requests directly to the server application.

Other Services

NFS and NIS are both RPC applications. Other RPC-based applications are included when NFS is installed on the system. A list and a short description follows:

Mount service	Handles NFS mount requests from NFS clients (**/usr/etc/rpc.mountd**).
Remote execution service	Remote execution daemon (**/usr/etc/rpc.rexd**, **/usr/btm/on**).
Remote System Statistics service	Obtains statistics about a remote system (**/usr/etc/rpc.rstatd**, **/usr/ucb/rup**).
Remote User Listing service	Lists the users on a remote system (**/usr/etc/rpc.rusersd**, **/usr/ucb/rusers**).

Boot Parameters service	Provides boot parameters to SunOS diskless clients (**/usr/etc/rpc.bootparamd**).
Remote Wall service	Remote wall service used by the **rwall** command (**/usr/etc/rpc.rwalld**, **/usr/etc/rwall**).
Spray service	Sends a one way stream of RPC packets (**/usr/etc/rpc.sprayd**, **/usr/etc/sprayspray**).
PC authentication service	Provides a user authentication service for PC-NFS (**/etc/rpc.pcnfsd**).

Related Information

The **exportfs** command, **mount** command, **smit** command, **umount** command.

The **biod** daemon, **mountd** daemon, **nfsd** daemon.

Alphabetical List of NFS Daemons and Commands on page 11–107, List of NFS Files on page 11–113, Functional List of NFS Daemons and Commands on page 11–110.

Network Information Service (NIS) Overview for System Management on page 11–47.

Using the Systems Management Interface Tool (SMIT) to Manage NFS and NIS on page 11–5, Using the System Resource Controller (SRC) to Manage NFS and NIS Daemons on page 11–6, Installing NFS and NIS on page 11–24, Configuring NFS On Your System on page 11–25, Maintaining NFS on page 11–31, Maintaining an NFS Server on page 11–31, Maintaining an NFS Client on page 11–34, Troubleshooting NFS–Related Problems on page 11–37, Administering Secure NFS on page 11–11.

Using the Systems Management Interface Tool (SMIT) to Manage NFS and NIS

The various management tasks for NFS are made available through the Systems Management Interface Tool (SMIT). SMIT may be used for such tasks as exporting directories, mounting file systems, rebuilding NIS maps, and assigning new keys to users for secure RPC applications.

The menus and dialogs that pertain to NFS and NIS can be reached through the `Communications Applications & Services` selection in the main menu of SMIT. The NFS menus can also be reached by selecting `Physical and Logical Storage` from the first menu, and then `Manage File Systems` from the second menu. The Manage File Systems menu contains an entry for NFS.

The NFS menu itself contains four items, as follows:

- `Configure TCP/IP (If Not Already Configured)`
- `Network File System (NFS)`
- `Network Information Service (NIS)`
- `Configure Secure NFS and NIS.`

The first entry, `Configure TCP/IP (If Not Already Configured)`, allows the person who administers the system to perform the necessary configuration of TCP/IP so that NFS or NIS can be used.

`Network File System (NFS)`, the second entry, leads to menus and dialogs that contain such NFS management tasks as starting or stopping NFS on the system, exporting directories to NFS clients, and mounting file systems from NFS servers.

`Network Information Service (NIS)`, the third entry, contains menus and dialogs that allow the person who administers the system to configure the host as an NIS server and client, to build maps and to transfer those maps to slave servers if they exist.

`Configure Secure NFS and NIS`, the fourth entry, is concerned with secure RPC applications. This menu has selections that allow the person who manages the system to start and stop the **keyserv** daemon, and to create or change keys for users.

Related Information

Network File System (NFS) Overview for System Management on page 11–1.
The **smit** command.
The **keyserv** daemon.

Using the System Resource Controller (SRC) to Manage NFS and NIS Daemons

The NFS and NIS daemons have been placed under the control of the System Resource Controller (SRC). This means that SRC commands such as **startsrc**, **stopsrc**, and **lssrc** should be used to manage NFS and NIS daemons, as shown in the following examples:

To start the **nfsd** daemons, enter:

```
startsrc -s nfsd
```

To start the collection of NFS daemons, enter:

```
startsrc -g nfs
```

Each of the NFS and NIS daemons has been defined in the database for SRC. SRC controls the NFS and NIS daemons. The SRC can start, stop, and obtain the status of daemons.

Some NFS daemons are not controlled by the SRC. The daemons that are started by the **inetd** daemon are not controlled by SRC.

The following is a list of the daemons and their subsystem names:

Daemons and their Subsystems		
File Path	**Subsystem Name**	**Group Name**
/usr/etc/nfsd	**nfsd**	nfs
/usr/etc/biod	**biod**	nfs
/usr/etc/rpc.lockd	**rpc.lockd**	nfs
/usr/etc/rpc.statd	**rpc.statd**	nfs
/usr/etc/rpc.mountd	**rpc.mountd**	nfs
/usr/etc/ypserv	**ypserv**	yp
/usr/etc/ypbind	**ypbind**	yp
/usr/etc/rpc.yppasswdd	**yppasswdd**	yp
/usr/etc/rpc.ypupdated	**ypupdated**	yp
/usr/etc/keyserv	**keyserv**	keyserv
/usr/etc/portmap	**portmap**	portmap

Each of these daemons can be selected with the SRC commands by either their subsystem name or the appropriate group name.

The above daemons do not support the long-listing facility of SRC nor the SRC trace commands.

The number of **biod** or **nfsd** daemons running on the system can be changed by using the **chnfs** command. This command allows the root user to control the number of daemons being executed by changing the appropriate entries in the SRC database.

Changing Command Line Arguments for Daemons Controlled by SRC

Many NFS and NIS daemons have command-line arguments that may be specified when the daemon is started. Since these daemons are not started directly from the command line, the SRC database must be updated so that the daemons are correctly invoked. The **chssys** command may be used to perform this task. For example:

```
chssys -s nfsd -a '10'
```

In this example the **nfsd** subsystem is changed so that when the daemon is started, the command line looks like `nfsd 10`. The changes made by the **chssys** command do not take effect until the subsystem has been stopped and restarted.

Related Information

The **chnfs** command, **lssrc** command, **startsrc** command, **stopsrc** command, **chssys** command.

The **inetd** daemon.

The **/etc/inetd.conf** file.

Network File System (NFS) Overview for System Management on page 11–1.

System Resource Controller Overview in *General Concepts and Procedures*.

Using Diskless Support for Sun3s and Sun4s Running SunOS 4.0

The NFS software contains support for Sun Diskless machines. The **/usr/etc/install** directory contains scripts for installing Sun distribution tapes on an AIX Version 3 machine. The scripts install the Sun executables in the requested directories, add entries in the **/etc/ethers** file for the diskless client, update the **/etc/bootparams** file with boot information, place a boot program in the **/tftpboot** directory for the diskless client, and update the **/etc/exports** file to include entries for the diskless client.

When this process is completed, the AIX Version 3 host is ready to act as a server for Sun diskless clients.

Two daemons need to be started by the **inetd** daemon. These are the **tftpd** daemons and the **bootparamd** daemon. The person who administers the system should make sure that the **inetd** daemon is configured correctly so that it starts these daemons when necessary. The entry for the **rpc.bootparamd** daemon should look like the following:

```
#bootparamd sunrpc_udp udp wait root /usr/etc/rpc.bootparamd
 bootparam 100026 1
```

This entry should be uncommented by removing the # (pound sign) and the **inetd** configuration database should be updated.

For more information on diskless support, see the **/usr/etc/install/README** file.

Related Information

Network File System (NFS) Overview for System Management on page 11–1.
The **bootparamd** daemon, **inetd** daemon, **tftpd** daemon.
The **/etc/inetd.conf** file.

Understanding NFS Access Control Lists (ACL) Support

NFS on AIX Version 3 has added support for access control lists (ACLs). This support is handled by an RPC program that exchanges information about ACLs between AIX Version 3 clients and servers. The ACL support does not change the NFS protocol specification; it is a separate function.

AIX Version 3 has added ACLs to the regular file system. Since the normal NFS protocol does not support ACLs, they cannot be seen by normal NFS clients. This results in unexpected behavior. A user on an NFS client might think that he has access to a file by looking at the permission bits, but the permissions could have been altered by the ACL associated with the file. Permissions on an AIX Version 3 server are enforced at the server according to the ACL on the server, so the user on the client machine could receive a permissions error.

When an AIX Version 3 client first attempts to access a remote mounted file system, it attempts to contact the ACL RPC program on the server.

If the server is an AIX Version 3 server, the Version 3 client consults the ACL associated with a file before granting access to the program on the client. This provides the expected behavior on the client when the request is sent over to the server. In addition, the **aclget**, **aclput**, and **alcedit** commands can be used on the client to manipulate ACLs.

Related Information

The **acledit** command, **aclget** command, **aclput** command.
Network File System (NFS) Overview for System Management on page 11–1.
Access Control Lists in *General Concepts and Procedures*.

Understanding NFS Mapped File Support

NFS mapped file support allows programs on a client to access a file as though it were memory. Using the **shmat** subroutine, users can map areas of a file into their address space. As a program reads and writes into this region of memory, the file is read into memory from the server or updated as needed on the server.

There are limitations to mapping files over NFS. First, files do not share information well between clients. Second, changes to a file on one client using a mapped file are not seen on another client. Third, locking and unlocking regions of a file is not an effective way to coordinate data between clients. If an NFS file is to be used for data sharing between programs on different clients, record locking and the regular **read** and **write** system calls should be used.

Multiple programs on the same client can share data effectively using a mapped file. Advisory record locking can coordinate updates to the file on the client. Multiple clients can share data using mapped files only if the data never changes; for example, as in a static database.

Related Information

Network File System (NFS) Overview for System Management on page 11–1.
The **read** subroutine, **shmat** subroutine, **write** subroutine.

Administering Secure NFS

NFS includes an authentication system that can be used by other UNIX and non-UNIX systems. The system uses Data Encryption Standard (DES) encryption and public key cryptography to authenticate both users and machines in the network. See Understanding DES Authentication for Secure NFS on page 11–11.

Public key cryptography is a cipher system that involves a public and a private key. The public key is published for general availability, while the private key is not. The private key is used to encrypt and decrypt data. The public and private keys are used to generate a common key, which is then used to create a DES key. See Understanding Public Key Encryption on page 11–13.

To create a secure file system over NFS, it is important to export directories securely and then mount file systems securely. In the server's **/etc/exports** file, the **–secure** option must be added as an export option to the directory entries that you want to secure. In the client's **/etc/filesystems** file, the **–secure** option must be added as a mount option to the file systems that you want to secure. If a client machine does not mount a secure file system with the **–secure** option, users have access as `nobody`, rather than as themselves.

The **/etc/publickey** file must contain an entry for each user. The entry includes the user's network name, a public key, and an encrypted secret key. The corresponding NIS **publickey.byname** map should reside only on the NIS servers.

The **/etc/publickey** file should exist on the NIS master server. Installation normally sets the only user as `nobody`. Users can establish their own public keys using the **chkey** command. The administrator can establish public keys for users using the **newkey** command. Because the NIS component must propagate a new map after each use of the **chkey** and **newkey** commands, only use these commands when a block of time is available on the system. Additionally, the **keyserv** daemon must be running. This daemon performs public key encryption and stores the private key in the **/etc/keystore** file. The **keylogin** command is used to provide keys to the keyserver. Do not delete the **/etc/keystore** file or the **/etc/.rootkey** file (the latter file contains the private key for a root user).

To keep passwords and private keys synchronized, maintain password entries with NIS and use the **yppasswd** command rather than the AIX **passwd** command to change passwords. The **yppasswd** command updates information in the **publickey** map when the password is changed.

If you re-install, move, or upgrade a machine, save the **/etc/keystore** file and the **/etc/.rootkey** file.

An advantage of the authentication system to secure file systems is the key exchange. This permits network-tapping problems to be handled on an individual basis.

Understanding DES Authentication for Secure NFS

There are two applications for DES authentication. The first application of DES authentication is a generalized NIS update service to update private fields in NIS maps. The **publickey** NIS map employs the DES-based update service. The second application of DES authentication is a process that places an encrypted time stamp in the credential and authenticates machines

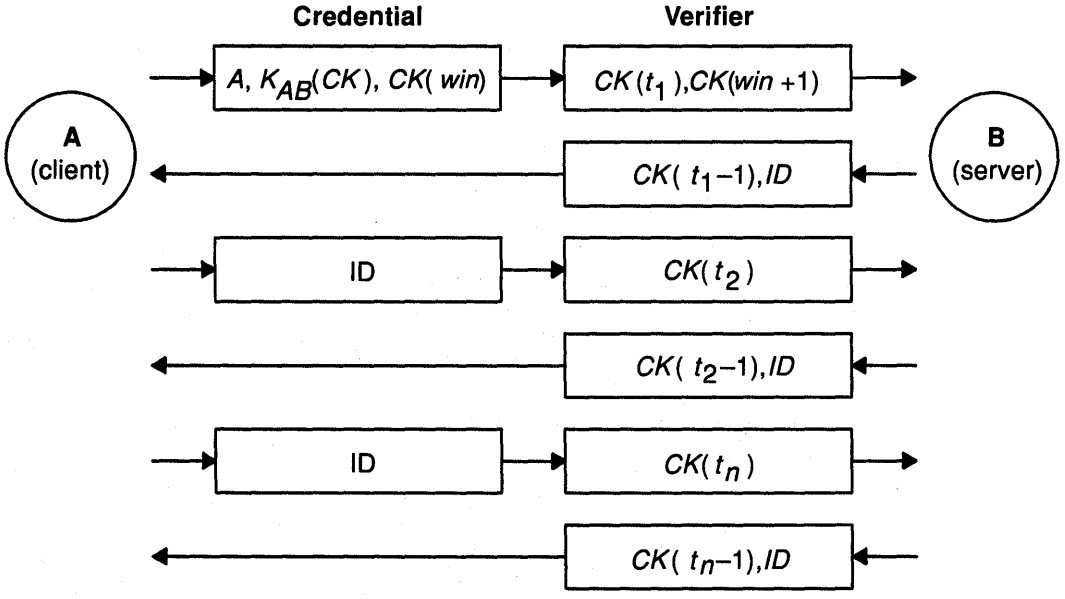
The level of security associated with each file system may be altered by the administrator. The **/etc/exports** file contains a list of file systems and the machines that can mount them. By default, file systems are exported with UNIX authentication, but the administrator can have them exported with DES authentication by specifying **–secure** on any line in the **/etc/exports** file.

The security of DES authentication is based on a sender's ability to encrypt the current time, which the receiver can then decrypt and check against its own clock. The time stamp is encrypted with DES. Two things are necessary for this scheme to work:

- The two agents must agree on the current time.
- The sender and receiver must be using the same encryption key.

If a network has time synchronization, then client/server time synchronization is performed automatically. If not, time stamps can be computed using the server's time instead of network time. In order to do this, the client asks the server what time it is before starting the RPC session, and then computes the time difference between its own clock and the server's. This difference is used to offset the client's clock when the server computes time stamps. If the client and server clocks get out of sync to the point where the server begins rejecting the client's requests, the DES authentication system resynchronizes with the server.

To get the same encryption key, when a client wants to talk to a server, it generates at random a key used for encrypting the time stamps (among other things). This key is known as the *conversation key* (*CK*). The client encrypts the conversation key using a public key scheme and sends it to the server during the first transaction. This key is the only thing ever encrypted with public key cryptography. For any two agents A and B, there is a DES key that only A and B can deduce. This key is known as the *common key* See the figure for an illustration of the authentication protocol.



The figure describes client A talking to server B. A term of the form *K*(*x*) means *x* encrypted with the DES key *K*. For its first request, the client's credential contains three things: the client's name, *A*; the conversation key *CK* encrypted with the common key; and the variable called *win* (window) encrypted with *CK*.

For secure NFS file systems, the window currently defaults to 30 minutes. The client's verifier in the first request contains the encrypted time stamp and an encrypted verifier of the specified window, *win* + 1. The window verifier makes guessing the right credential much more difficult, and increases security.

After authenticating the client, the server stores four things into a credential table: the client's name, *A*; the conversation key *CK*; the window; and the time stamp. The server only accepts time stamps that are chronologically greater than the last one seen, so any replayed transactions are guaranteed to be rejected. The server returns to the client in the verifier an index ID into the credential table, plus the client's time stamp minus one, encrypted by *CK*. The client knows that only the server could have sent such a verifier, since only the server knows what time stamp the client sent. The reason for subtracting one from it is to ensure that it is not valid and cannot be reused as a client verifier.

After the first transaction, the client just sends its ID and an encrypted time stamp to the server, and the server sends back the client's time stamp minus one, encrypted by *CK*.

Understanding Public Key Encryption

The particular public key encryption scheme is the Diffie-Hellman method. This algorithm works by generating the secret key at random and computing a public key using the following formula (*PK* and *SK* are 128-bit numbers):

$$PK_A = \alpha^{SK_A}$$

The public key is stored in a public directory, but the secret key is kept private. Next, public key is generated from secret key in the same manner as above. Now the common key can be derived as in the following formula:

$$K_{AB} = PK_B^{SK_A} = (\alpha^{SK_B})^{SK_A} = \alpha^{(SK_A\ SK_B)}$$

Without knowing the client's secret key, the server can calculate the same common key in a different way; see the following formula:

$$K_{AB} = PK_A^{SK_B} = (\alpha^{SK_A})^{SK_B} = \alpha^{(SK_A\ SK_B)}$$

Only the server and client can calculate this common key since doing so requires knowing one secret key or the other. The formula is computed modulo *M*, another well-known constant. Outsiders cannot guess your secret key by taking the logarithm of your public one because the size of *M* makes it a computationally infeasible task. To be secure, the common key has too many bits to be used as a DES key, so 56 bits are extracted from it to form the DES key.

Both the public and the secret keys are stored indexed by net name in the NIS map **publickey.byname**. The secret key is DES-encrypted with your keylogin password.The **keylogin** command uses your encrypted secret key, decrypts it with your login password, and gives it to a secure local keyserver to save for use in future RPC transactions. Ordinary users do not have to be aware of their public and secret keys. In addition to changing your keylogin password, the **yppasswd** program generates a new public/secret key pair.

The **keyserv** keyserver is an RPC service local to each machine that performs the three public key operations, as follows:

```
setsecretkey(secretkey)
encryptsessionkey(servername, des_key)
decryptsessionkey(clientname, des_key)
```

The `setsecretkey` operation tells the keyserver to store away your secret key (SK_A) for future use; it is normally called by the **keylogin** command. The client program calls the `encryptsessionkey` operation to generate the encrypted conversation key that is passed in the first RPC transaction to a server. The keyserver looks up *`servername`*'s public key and combines it with the client's secret key (set up by a previous `setsecretkey` call) to generate the key that encrypts `des_key`. The server asks the keyserver to decrypt the conversation key by calling the `decryptsessionkey` operation.

Implicit in these procedures is the name of the caller, who must be authenticated in some manner. The keyserver cannot use DES authentication to do this, since it would create deadlock. The keyserver solves this problem by storing the secret keys by the UID and only granting requests to local root processes. The client process then executes a **setuid** process, owned by the root user, which makes the request on the part of the client, telling the keyserver the real UID of the client. Ideally, the three operations described previously would be system calls, thus allowing the kernel to contact the keyserver directly instead of executing the **setuid** program.

Naming Network Entities for DES Authentication

DES authentication bases naming upon new names called net names. A net name is a string of printable characters to authenticate. The public and secret keys are stored on a per-net name, rather than a per-user name, basis. The **netid.byname** NIS map maps the net name into a local UID and group-access list.

User names are unique within each domain. Net names are assigned by concatenating the operating system and user ID with the NIS and Internet domain names. A good convention for naming domains is to append the Internet domain name (COM, EDU, GOV, MIL) to the local domain name.

Network names are assigned to machines as well as to users. A machine's net name is formed much like a user's. For example, an AIX machine named `hal` in the `eng.ibm.COM` domain has the net name `unix.hal@eng.ibm.com`. Proper authentication of machines is important for diskless machines that need full access to their home directories over the network.

To authenticate users from any remote domain, make entries for them in two NIS databases. One is an entry for their public and secret keys, the other is for their local UID and group-access list mapping. Users in the remote domain can then access all of the local network services, such as the NFS and remote logins.

Booting Considerations of Public Key Systems

When a machine reboots after a power failure, all of the stored secret keys get wiped out, and no process can access secure network services, such as mounting an NFS file system. Root processes could continue if there were someone to enter the password that decrypts the root user's secret key. The solution is to store the root user's decrypted secret key in a file that the key server can read.

Not all **setuid** programs behave as they should. For example, if a **setuid** program is owned by `dave`, who has not logged into the machine since it booted, then the program cannot access any secure network services as `dave`. Most **setuid** programs are owned by the root user, and the root user's secret key is always stored at boot time.

Performance Considerations of Public Key Systems

Public key systems can be slow. However, public key encryption occurs only in the first transaction with a service, and caching speeds the process. The first time a client program contacts a server, both the client and the server must calculate the common key.

The time it takes to compute the common key is the time it takes to compute an exponential modulo *M*: roughly 1 second. This means it takes 2 seconds just to get things started, since both client and server have to perform this operation. You have to wait only the first time you contact a machine. Since the keyserver caches the results of previous computations, it does not have to recompute the exponential every time.

Secure NFS is the most important service in terms of performance. The extra overhead that DES authentication requires versus UNIX authentication is the encryption. A time stamp is a 64-bit quantity, the same size as the DES block size. Four encryption operations take place in an average RPC transaction: the client encrypts the request time stamp, the server decrypts it, the server encrypts the reply time stamp, and the client decrypts it. If you use secure networking, there is a performance hit that impacts network performance. Since not all file operations go over the wire, the impact on total system performance is lower than this. And because security is optional, environments that require higher performance can turn it off.

Related Information

The **chkey** command, **keylogin** command, **newkey** command, **passwd** command, **yppasswd** command.

The **keyserv** daemon.

The **key_decryptsession** subroutine, **key_encryptsession** subroutine, **key_setsecret** subroutine.

How to Use the NFS Secure Option When Exporting a Directory on page 11–78, How to Use the NFS Secure Option When Mounting a File System on page 11–85.

Configuring Secure NFS on page 11–16, Maintaining Secure NFS on page 11–18.

Network File System (NFS) Overview for System Management on page 11–1.

Understanding RPC Authentication in *Communications Programming Concepts.*

Configuring Secure NFS

In order to use secure NFS, NIS needs to be configured and running on your network. The NIS **publickey** and **netid** maps must be set up and the NIS daemons which access these maps must be running.

The NIS master server must be running the **keyserv**, **yppasswdd**, and **ypupdated** daemons. All NIS clients must be running the **keyserv** daemon (in addition to the daemons needed for normal NIS operations).

Since passwords are an important factor in forming the keys in the **publickey** map, it is a good idea (and strongly recommended) to set up the **yppasswdd** daemon and use the **yppasswd** command to update passwords on your systems. The **yppasswdd** daemon updates the password in the **publickey** map so users only need to remember one password. This also ensures that the passwords remain in sync with each other.

Remember that in order to create a secure NFS environment, you must export directories as well as mount them with the secure option.

Start the **keyserv** daemon on the master server as well as the slave servers and clients by uncommenting the following stanza in the **/etc/rc.nfs** file.

```
#if [ -x /usr/etc/keyserv ]; then
#
#fi
```

Note: Remove the # (pound sign) from the beginning of the lines.

You also need to start the **ypupdated** daemon on the master server by uncommenting the following stanza in /etc/rc.nfs.

```
#if [ -x /usr/etc/rpc.ypupdated -a -d /etc/yp/'domainname' ]; then
#
#fi
```

Note: Remove the # (pound sign) from the beginning of the lines.

Finally, you should start the **yppasswdd** daemon on the master server by uncommenting the following stanza in the **/etc/rc.nfs** file.

```
#DIR=/etc/passwd
#if [ -x /usr/etc/rpc.yppasswdd -a -f $DIR/passwd ]; then
#fi
```

Note: Remove the # (pound sign) from the beginning of the lines.

Start the daemon by issuing the following command:

```
start src -s yppasswdd
```

Related Information

The **chkey** command, **newkey** command, **yppasswd** command.

The **keyserv** daemon, **yppasswdd** daemon, **ypupdated** daemon.

How to Change NIS Passwords on page 11–97, How to Configure NIS on page 11–71, How to Use the NFS Secure Option When Exporting a File System on page 11–78, How to Use the NFS Secure Option When Mounting a File System on page 11–85.

Administering Secure NFS on page 11–11, Changing Security with NIS on page 11–56.

Network File System (NFS) Overview for System Management on page 11–1, Network Information Service (NIS) Overview for System Management on page 11–47.

Maintaining Secure NFS

Maintaining a secure network involves the normal procedures for maintaining NIS and NFS. In addition, it is important to make sure that the daemons needed for maintaining the **publickey** map (**yppasswdd** and **ypupdated**) are always running on the NIS master server. Also, the **keyserv** daemon should be running on all machines using secure NFS. It is important to make sure the passwords are updated using the **yppasswd** command, since the **yppasswdd** daemon updates keys in the **publickey** map when passwords are changed (keys are based on passwords as well as times).

Related Information

The **yppasswd** command.

The **keyserv** daemon, **yppasswdd** daemon, **ypupdated** daemon.

How to Change NIS Passwords on page 11–97.

Maintaining NIS on page 11–54, Maintaining NFS on page 11–31.

Network File System (NFS) Overview for System Management on page 11–1, Network Information Service (NIS) Overview for System Management on page 11–47.

Understanding the Network Lock Manager

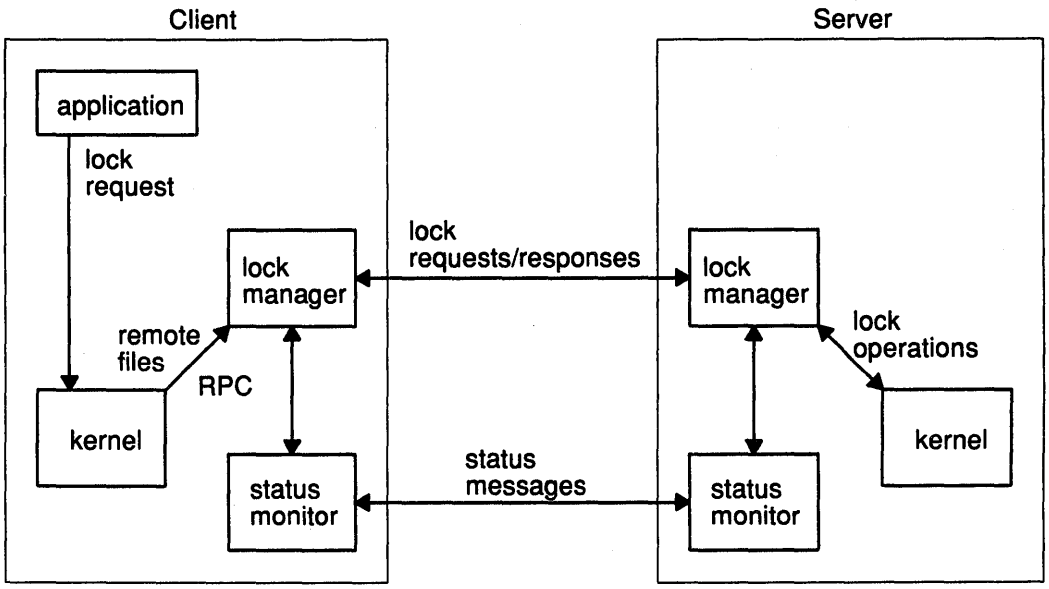
The NFS-compatible Network Lock Manager supports the System V style of advisory file and record locking over the network. The services are accessed with the **lockf**, **fcntl**, and **flock** system calls. The **flock** function is implemented with the **fcntl** function.

When a lock is attempted on a file in an NFS-mounted directory, the kernel makes a local RPC request to the network lock manager daemon (**rpc.lockd**). This daemon then makes a lock request to the network lock manager daemon on the NFS server. Lock requests for local files do not use the **rpc.lockd** daemon; they are handled by the kernel instead.

The Locking Service Structure

The network lock manager contains both server and client functions. The client functions are responsible for processing requests from the kernel and sending requests to the network lock manager at the server. The server functions process lock requests from the network and perform lock operations in the kernel.

The following figure depicts the overall architecture of the locking service.



When an application makes a lock request for a local file, the kernel processes the lock request. However, if the lock request is for a remote file, the kernel forwards the request to the lock manager. The lock manager then sends the request to the lock manager on the server. The lock manager on the server requests the lock from its kernel and sends the reply back to the client. The lock manager registers its interactions with other machines by using the Network Status Monitor (**rpc.statd**). The status monitor watches the status of registered machines and notifies the lock manager when one of them is rebooted.

At each server the lock manager process accepts lock requests that are made on behalf of client processes by a remote lock manager or on behalf of local processes by the kernel. The client and server lock managers communicate using RPC calls. When the lock manager receives a remote lock request for a machine on which it does not already hold a lock, the lock manager registers its interest in that machine with the local status monitor and then waits for the monitor to notify it when the machine is up.

The monitor continues to watch the status of registered machines and notifies the lock manager if one of them is restarted (after a crash, for instance). The lock manager tries the lock operation in the kernel and communicates back to the client's **lockd** daemon.

The crash recovery procedure is simple. If the failure of a client is detected, the server releases the failed client's locks, on the assumption that the client application will request locks again as needed. If the crash and recovery of a server is detected, the client lock manager retransmits all lock requests previously granted by the server. This retransmitted information is used by the server to reconstruct its locking state during a grace period.

The locking service is thereby essentially stateless. The information about the state of a server is carefully circumscribed within a pair of system daemons that are set up for automatic, application-transparent crash recovery. If a server crashes and loses its state, the server expects its clients to be notified of the crash (when the server recovers) and to send the server necessary information to reconstruct its state. Reconstructing the former state can be achieved using information from the network status monitor, which the lock manager uses to detect both client and server failures.

Locking and Recovery

In an NFS environment, where multiple machines can have access to the same file at the same time if a computer fails, the process of recovering is necessarily more complex than in a non-network environment. Furthermore, locking is *inherently stateful.* If a server crashes, clients with locked files must be able to recover their locks. If a client crashes, its servers must hold the client's locks while it recovers.

Additionally, to preserve the overall transparency of NFS, the recovery of lost locks must not require the intervention of the applications themselves. This can be accomplished as follows:

- Basic file access operations, such as read and write, use a stateless NFS protocol. All interactions between NFS servers and clients are atomic; that is, the server does not retain information on clients from one interaction to the next. In case of a server crash, client applications simply sleep until the server comes back up and their NFS operations can complete.
- Stateful services, such as the locking service, are not part of NFS. These services require the server to maintain client information from one transaction to the next. They are separate services that use the Network Status Monitor to ensure that their implicit network state information remains consistent with the real state of the network. There are two specific state-related problems involved in providing network locking:
  - If the client fails, its locks are held by the server until the client restarts, at which time the server releases them.
  - If the server fails, it loses its state information as well as all of its locks. Upon restart, the **lockd** daemon allows clients to reclaim their locks during a grace period.

  The Network Lock Manager solves both of these problems by cooperating with the Network Status Monitor to ensure that it is notified of relevant machine crashes. The lock manager's protocol then allows it to recover the lock information it needs when crashed machines recover.

The Network Lock Manager and the Network Status Monitor are network-service daemons, which means they run at user level. However, both daemons are essential to the kernel's ability to provide fundamental network services, and they are therefore run on all network machines.

Like other network service daemons (such as the **rexd** daemon, which provides remote execution services, and the **rlogind** daemon, which provides remote login services), the lock manager and status monitor daemons are seen as extensions to the kernel. These extensions are implemented as daemons.

The Network Status Monitor is used by the Network Lock Manager to maintain the inherently stateful locking service within the stateless NFS environment.

The status monitor provides a general framework for collecting network status information. As a daemon that runs on all network machines, the status monitor implements a simple protocol that allows the Network Lock Manager to monitor the status of other machines. Use of this protocol can avoid situations in which the Network Lock Manager, running on different machines (or even on the same machine) disagrees about the status of a site. Such situations can cause inconsistencies in the Network Lock Manager.

To use the status monitor, the Network Lock Manager registers with it the machines that the network is interested in. The monitor then tracks the status of those machines. When one of them crashes, the monitor notifies the Network Lock Manager. The Network Lock Manager can then take whatever actions are necessary to re-establish a consistent state.

Related Information

Configuring the Network Lock Manager on page 11–21, Maintaining the Network Lock Manager on page 11–22, Troubleshooting the Network Lock Manager on page 11–23.

Network File System (NFS) Overview for System Management on page 11–1.

Configuring the Network Lock Manager

To configure the Network Status Monitor:

- TCP/IP must be installed and configured.
- NFS must be installed.
- NFS servers and clients must be configured.

When you start NFS, the **rpc.statd** and **rpc.lockd** daemons should have been started for you through the **/etc/rc.nfs** file. If NFS is already up and running, you can check this by typing **ps –ef** and looking for the **rpc.lockd** and **rpc.statd** daemons. If for some reason you don't see both of these daemons up and running, you need to check the following items.

Make sure the **rpc.statd** and **rpc.lockd** entries are uncommented in the **/etc/rc.nfs** file. Edit the **/etc/rc.nfs** file and search for the following lines:

```
if [ –x /usr/etc/rpc.statd ]; then
       startsrc –s rpc.statd
fi

if [ –x /usr/etc/rpc.lockd ]; then
       startsrc –s rpc.lockd
fi
```

If pound signs (#) are present at the beginning of any of these lines, delete them so the entries resemble the ones above. After you have confirmed this, you can start the **rpc.statd** and **rpc.lockd** daemons by entering the following:

```
startsrc –s rpc.statd
startsrc –s rpc.lockd
```

If NFS is up and running and the entries in the **/etc/rc.nfs** file look correct, then you can restart the daemons by hand. You can do this by the following:

```
stopsrc -s rpc.lockd
stopsrc -s rpc.statd
startsrc -s rpc.statd
startsrc -s rpc.lockd
```

If for some unknown reason this does not get the daemons up and running, refer to the section on Troubleshooting the Network Lock Manager on page 11–23.

Related Information

How to Configure NFS on Your System on page 11–63, How to Configure Clients and Servers on page 11–67, How to Configure an NFS Server on page 11–66, How to Configure an NFS Client on page 11–65.

Understanding the Network Lock Manager on page 11–19 .

Maintaining the Network Lock Manager

Once you have installed and configured NFS, the locking daemon itself requires no real maintenance other than getting things back to normal in the event of a malfunction. If you are having any problems in getting a lock from a client, make sure that:

- The **rpc.statd** daemon is running (on client and server)
- The **rpc.lockd** daemon is running (on client and server).

If these daemons are running (or appear to be), things should work as normal. In the event of a malfunction, you might be required to periodically reset these daemons on clients and servers. These steps are given under the assumption that all applications are terminated or are in a state of no return. On the server and the client, stop the **lockd** and **statd** daemons by entering:

```
stopsrc -s rpc.statd
stopsrc -s rpc.lockd
```

Remove the **/etc/sm** and **/etc/rm.bak** directories along with the **/etc/state** file by entering:

```
rm -rf /etc/sm /etc/sm.bak /etc/state
```

This ensures that there is no previous state information held by the client. Now you are ready to restart the daemons. On the server and client, enter:

```
startsrc -s rpc.statd
startsrc -s rpc.lockd
```

Note: Order is important here. Always start the **statd** daemon first.

When restarting the **statd** daemons, you might get a message that resembles the following:

```
clnttcp_create: RPC: Remote System error - Connection refused
rpc.statd:cannot talk to statd at {server}
```

If you start getting these messages on any client, whether it be after a restart or not, you need to verify that the daemons on the server are up and running. This error is directly related to the functions of the **/etc/sm** file or files. You can look in the **/etc/sm** directory to see which machines the **statd** daemon expects to communicate with. When a **statd** daemon expects a connection with a server to be active and it is not, the daemon produces the above diagnostic message on the console.

This message does not imply that a client is trying to reclaim locks, only that the status monitor believes that the server is in a state contrary to what its **/etc/sm** file indicates.

After you have restarted the daemons, remember that there is a grace period (default of 45 seconds). During this time, the **lockd** daemons allow reclaim requests to come from other clients that previously held locks with the server, so you will not get a new lock immediately after starting the daemons.

Related Information

The **startsrc** command, **stopsrc** command.
The **lockd** daemon, **statd** daemon.
How to Configure NFS on Your System on page 11–63, How to Set Up NFS Servers and Clients on page 11–67, How to Configure an NFS Server on page 11–66, How to Configure an NFS Client on page 11–65.
Configuring the Network Lock Manager on page 11–21, Troubleshooting the Network Lock Manager on page 11–23, Understanding the Network Lock Manager on page 11–19.

Troubleshooting the Network Lock Manager

When you get to a point where none of the above maintenance operations seem to work, you can run the **lockd** daemon in debug mode, so that you can actually verify what it is doing. This should only be done in cases where nothing else has worked and you suspect a fatal problem with the **lockd** daemon itself, on the server or the client. To run the **lockd** daemon in debug mode, open a shell in another window or terminal and start it with the **–d1** flag:

```
/usr/etc/rpc.lockd –d1
```

When invoked this way, the **lockd** daemon provides diagnostic messages on standard output. At first, there will be a number of messages dealing with the grace period; wait for them to time out. After the grace period has timed out on both the server and any clients, you can run your application that is performing the locks in question and verify that the request is being transmitted from client to server and server to client. It is handy to have a pair of terminals side by side for this.

Related Information

The **lockd** daemon.
How to Configure NFS on Your System on page 11–63, How to Configure Clients and Servers on page 11–67, How to Configure an NFS Server on page 11–66, How to Configure an NFS Client on page 11–65.
Configuring the Network Lock Manager on page 11–21, Maintaining the Network Lock Manager on page 11–22, Understanding the Network Lock Manager on page 11–19.

Installing NFS and NIS

Network File System (NFS) is shipped as part of the Base System Network Package. The installation procedures conform to the standards and conventions for all AIX licensed programs. Be sure no other system activity is going on when the program is being installed on your system.

Installing NFS for the First Time

The NFS component is installed with the network information service (NIS) component. The NIS component can be installed and used without using NFS.

Installation of the NFS component makes it possible to use computers as NFS servers and clients. During installation, the NFS kernel extension is placed in the **/etc/nfs.ext** file. As a kernel extension, NFS is loaded into the kernel dynamically when the **/etc/rc.net** script is executed. This alleviates the need to rebuild the kernel upon installation.

Installation of the NIS component enables the system to consolidate information onto a centralized map and then use the network information service (NIS) to administer the information from the map.

Installation of both the NFS and NIS components at the same time causes the components to be added to the appropriate places and the kernel to be rebuilt.

Install and configure NFS by setting up and configuring each system on the network to be either a server, a client, or both.

Reinstalling NFS

If you re-install NFS, the system backs up the **/etc/rc.nfs** file. You may have altered this file. The **/usr/lpp/nfs** directory serves as the root directory for the backup. The NFS install program reconfigures the **/etc/rc.nfs** file exactly as if it were installing NFS for the first time.

Related Information

The **/etc/vfs** file.
List of NFS Files on page 11–113.
Network File System (NFS) Overview for System Management on page 11–1.
How to Install AIX for RISC System/600 Licensed Program Optional Program Products in *General Concepts and Procedures.*
Configuring NFS on Your System on page 11–25, Maintaining NFS on page 11–31.

Configuring NFS On Your System

Determine the NFS network configuration before beginning. Planning a configuration involves determining which systems in the network should be set up as servers and which should be set up as clients. A system can be configured as both a server and a client.

Note: Only root users can configure NFS.

Some of the tasks for configuring NFS apply to both servers and clients, while other steps apply only to one or the other. The tasks for configuring NFS are:

- Configuring an NFS server
- Configuring an NFS client
- Configuring NFS server and clients
- Configuring PC-NFS, if necessary.

The specific tasks for configuring a server or configuring a client should be completed before doing the tasks that apply to both the server and client.

These NFS configuration tasks are explained in detail in the following paragraphs.

Configuring an NFS Server

Once it is determined which systems will be NFS servers, create an **/etc/exports** file to allow the server to export directories.

Exporting Directories

There are two ways to control how a server exports directories:

- Modifying the **/etc/exports** file
- Using the **exportfs** command.

The /etc/exports File

The **/etc/exports** file advertises all directories that a server exports to its clients. Only exported directories can be mounted by clients. Modifying the **/etc/exports** file allows a server to control which clients can mount a directory by limiting access to the directory to a desired client or netgroup.

The following lines illustrate example entries from an **/etc/exports** file:

```
/u              -root=ballet,access=ballet
/usr/games      -ro,access=ballet:jazz:tap
/usr/tmp
/usr/lib        -access=clients
```

The syntax in the examples is:

```
directory       -options[,option]
```

The `directory` is the full path name of the directory which is the local mount point. Options may designate a simple flag such as `ro` or a list of host names.

The following lines illustrate what the above examples mean in terms of the exports. See the specific documentation of the **/etc/exports** file and the **exportfs** command for a complete list of options and their descriptions.

```
/usr/games      -ro,access=ballet:jazz:tap
```

The previous example means that the `/usr/games` directory can be mounted by the systems named `ballet`, `jazz`, and `tap`. They cannot write in the directory but can read data and execute programs from the directory.

```
/u              -root=ballet,access=ballet
```

The foregoing example means that the `/u` directory can be mounted by the system `ballet` and that root access is allowed for the directory.

```
/usr/tmp
```

The absence of an access list in the previous example means that any client can mount this directory.

```
/usr/lib        -access=clients
```

The foregoing is an example of an access list designated by specifying a netgroup. A netgroup is a network-wide group allowed access to certain network resources for security or organizational purposes. Machines designated as belonging to the netgroup clients can mount the **/usr/lib** directory from this server. Netgroups are controlled by using NIS and by editing the **/etc/netgroup** file. NIS must be used to do the netgroup mapping.

Creating the /etc/exports File

The server's **/etc/exports** file lists the directories that are available for mounting by clients. The server automatically exports the listed directories each time the NFS server is started. The **/etc/rc.nfs** script does not start the **nfsd** daemons or the **rpc.mountd** daemon if the **/etc/exports** file does not exist.

The **/etc/exports** file may be created by a text editor. It is also created by using the **mknfsexp** command. See How to Configure an NFS Server on page 11–66 for more details on the creation of the **/etc/exports** file. If editing the **/etc/exports** file, list each directory to be exported starting in the left margin. The path names can be any fully qualified path name, but may not include any directories that are already remote mounts. If controlling client access is desired, then use the **–access** option. To make the modified entries take effect, use the **exportfs** command. It should be noted that a server can only export its own directories. A server cannot act as a messenger between clients and the directories of other servers.

Configuring an NFS Client

After at least one NFS server is configured, the configuration of each system designated as an NFS client can proceed. Configuring an NFS client involves establishing NFS as the default remote file system, starting the NFS daemons, and establishing the local mount points and mounts.

The tasks that must be performed to configure each system designated as an NFS client are:

- Making NFS the default remote file system
- Starting the NFS daemons
- Using the mkdir command to establish the local mount points
- Mounting the desired directories.

These client configuration tasks are explained in the following paragraphs.

Making NFS the Default Remote File System

To make NFS the default remote file system, use an editor to search for and uncomment the following lines in the **/etc/vfs** file:

```
#%defaultvfs jfs nfs

#nfs 2 /etc/helpers/nfsmnthelp none remote
```

If a # (pound sign) appears at the beginning of the lines as shown, delete the pound signs. This lets the **mount** command determine the remote file system type by recognizing the following syntax:

```
mount remotehost:/remote/directory /Local/Directory
```

If the **/etc/vfs** file is not edited in this way, the **–v** option of the **mount** command must be used. For example:

```
mount –v nfs remotehost:/remote/directory /Local/Directory
```

Starting the NFS Daemons

The NFS daemons, by default, are not started. When NFS is installed, all of the files are placed on the system but the steps to activate NFS are not taken. When the system starts, the **/etc/rc.nfs** file should be executed to start the NFS daemons. This action can be made to happen on system restart by placing an entry in the **inittab** file. The **mknfs** command supplied with NFS does all of the required setup to have NFS started by default. There is a SMIT menu for this command that can be reached by executing the **smit mknfs** command.

The user has the choice of starting NFS at any time, only at system restart, or both. If NFS is started immediately, the **/etc/rc.nfs** file is executed to start the NFS daemons. If NFS is started at system restart, an entry is placed in the **inittab** file so that the **/etc/rc.nfs** file is executed each time the system restarts. If the choice of both is taken, then an entry for the **inittab** file is created and the **/etc/rc.nfs** file is executed immediately.

Establishing Local Mount Points

For NFS to complete a mount successfully, a directory that will act as the mount point of an NFS mount must be present. This directory should be empty. It acts as the placeholder for the NFS mount that is to occur. This mount point can be created like any other directory with the **mkdir** command. There are no special attributes needed for this directory.

Mounting the Desired File Systems

After the NFS client is configured, the final step is to mount a file system from a server. There are three kinds of mounts:

- Predefined mounts
- Explicit mounts
- Automatic mounts.

Predefined mounts involve updating the **/etc/filesystems** file to add a stanza. This stanza or entry defines the characteristics of the mount. Data such as the host name, remote path, local path, and any mount options are listed in this stanza.

Explicit mounts are usually done for short periods of time for special purposes. These mounts are usually fully qualified on the command line by using the **mount** command with all needed information. Explicit mounts do not require updating the **/etc/filesystems** file. If the system is restarted when one of these mounts exist, the mount is not reestablished when NFS is started.

Automatic mounts are controlled by the **automount** daemon. This daemon monitors specified directories for activity. If a directory is needed and it is supposed to be remotely mounted, the **automount** daemon intercepts the request, mounts the file system, and then services the request.

Each type of mount has its purpose and special configuration. See How to Mount a File System Using NFS on page 11–80 for more details on how to establish these mounts.

Configuring NFS Servers and Clients

After determining which systems will be clients, which servers, and which both, complete the procedures that apply exclusively to a server or a client. Both servers and clients start the NFS daemons, and both servers and clients update the **inetd** daemon's configuration database. All of these procedures are explained below.

There are three ways that NFS can be started on a system: a SMIT interface that starts NFS, the **mknfs** command that starts NFS, and a procedure for editing appropriate files to start NFS.

To use the SMIT interface to start NFS, the user should execute the **smit mknfs** command. This presents the user with a menu that, when committed, starts NFS. When NFS is started this way, an entry is added to the **inittab** file. The **inittab** file keeps track of those things that are to be executed when the system restarts. The entry that is added for NFS executes the **/etc/rc.nfs** file. This file in turn starts all of the needed NFS daemons for a particular system. The **portmap** daemon is also started; the entry in the **/etc/rc.tcpip** file is uncommented (or activated) so that when that file is executed before the **/etc/rc.nfs** file, the **portmap** daemon is available for the NFS daemons.

The next method that can be used to start NFS is to execute the **mknfs** command. This command does the same thing as the SMIT interface. In fact, the SMIT interface executes the **mknfs** command to do the work of starting NFS.

The last method of starting NFS is to edit all of the appropriate files. As mentioned, the **/etc/rc.tcpip** file is updated so that the **portmap** daemon is started and an entry needs to be placed in the **inittab** file. The **inittab** entry can be made by using the **mkitab** command. The entry for the **/etc/rc.nfs** file in the **inittab** file should be very similar to the entry for the **/etc/rc** file. After those two things are done, the **/etc/rc.nfs** file should be checked to make sure that the NFS daemons are started when the file is executed.

The **inetd** daemon has a textual representation of its configuration database in the **/etc/inetd.conf** file. If this file is updated directly to activate or deactivate entries, the effects do not take place until the **inetimp** command is executed and the **inetd** daemon is refreshed. Since the **inetd** daemon is under the control of the **srcmstr** daemon, the command needed to refresh the **inetd** daemon is:

```
refresh -s inetd
```

This causes the **inetd** daemon to reread its configuration database. For a full description of how to update the database see How to Configure the inetd Daemon on page 14–102. The reason to change this configuration is that there are many RPC-based daemons that are started by the **inetd** daemon. These daemons are installed with NFS and are by default not active.

Configuring PC-NFS

PC-NFS permits personal computers to communicate with NFS servers. If your network includes IBM Personal Computers that are running versions of the Disk Operating System (DOS) and have the PC-NFS program installed, you can configure the **pcnfsd** daemon on network servers to allow this communication. PC-NFS permits access to authentication services and print-spooling services.

The tasks that must be performed to configure PC-NFS are:

- Editing the **/etc/inetd.conf** file to start the **pcnfsd** daemon
- Using the **rpcinfo** command to verify the **pcnfsd** daemon is accessible.

Without the **pcnfsd** daemon, users on PC-NFS client systems can perform many PC-NFS functions that do not involve specialized server support, such as mounting exported file systems and requesting network addresses and host names. However, access to authentication and print-spooling services is not possible without the **pcnfsd** daemon.

Before configuring the **pcnfsd** daemon, select a suitable location for the spool directories, such as the **/usr/tmp** directory. The spool directory must have at least 100K bytes of free space. Next, export the file system containing the spool directories. Do not put access restrictions on the exported file system that might cause problems in your network.

You should configure the **pcnfsd** daemon on the following:

- Systems that perform user authentication services
- Systems that offer print-spooling
- All NIS master and slave servers.

Note: Typical NIS networks are configured so that PC-NFS can pick any NIS server as the default server, so it is important that all servers have the **pcnfsd** program running. If running this program on all NIS servers is not practical, or if you want to limit requests to a specific server, add a **net pcnfsd** command to the **autoexec.bat** file on each personal computer to force it to use a specific NIS server.

Understanding PC-NFS Authentication Services

The authentication capabilities of the **pcnfsd** program enable the person who manages the system to monitor system resources and select the appropriate system security by recognizing individual users and assigning them different privileges. By default, PC-NFS presents itself to NFS servers as the user `nobody`. With `nobody` privileges, there is no way to tell which user has accessed a resource because all personal computer user files appear as owned by `nobody`.

When the **pcnfsd** daemon is installed, a PC-NFS user can issue the **net name** command from a personal computer to log in to PC-NFS in the same manner as the user can log in to AIX. The user's name and password are verified by the **pcnfsd** daemon. This authentication procedure does not make a server more secure, but it does give a user more control over access to files that are available through NFS.

Understanding PC-NFS Print-Spooling Services

When it starts, the **pcnfsd** daemon invokes the standard print-spooling subsystem, which allows it to be used with remote printers. When properly configured, PC-NFS redirects files intended for personal computer printers to a network print-spooling service. The **pcnfsd** process places the personal computer file in a spooling directory, and invokes the printing function with appropriate parameters. The directory used for spooling is passed as a value to the **pcnfsd** daemon. The directory must be in an exported file system that is accessible to PC-NFS clients.

To print, PC-NFS mounts the spool directory as an NFS virtual drive and creates spool files in the directory. PC-NFS calls the **pcnfsd** daemon with the following information:

- The name of the file to be printed
- The login ID of the user on the client
- The name of the printer to be used.

The **pcnfsd** daemon then invokes the local spooler to print the file.

Editing the /etc/inetd.conf File to Start the pcnfsd Daemon

You can start the **pcnfsd** daemon in different ways, depending on whether you want to use the default spooling directory or another spooling directory. The spooling directories need to be cleared periodically. You can use the **rpcinfo** command to verify that the **pcnfsd** daemon is accessible.

Starting the pcnfsd Daemon from the Default Directory

You can use the **/usr/tmp** directory as the default spooling directory. To start the **pcnfsd** daemon, edit the **inetd.conf** file by uncommenting the **pcnfsd** entry. The next time the **inetd** daemon runs, **pcnfsd** print-spooling requests are sent to the **/usr/tmp** directory.

Note: You cannot change the name of the default directory in the **inetd.conf** file.

Starting the pcnfsd Daemon from a Different Directory

You can choose a directory other than the **/usr/tmp** default spooling directory for starting the **pcnfsd** daemon. This involves editing the **/etc/rc.nfs** file by adding an entry.

If you specify a print-spooling directory other than the default, be sure to deactivate the **pcnfsd** print-spooling default directory by commenting its entry in the **inetd.conf** file.

Since printer redirection requests sometimes cause file listings of zero length to be left in the PC-NFS spool directories, you should periodically clear spooling directories of these entries. You can create a shell script that removes the zero-length files and have the **cron** process execute it at appropriate intervals.

Using the rpcinfo Command to Verify the pcnfsd Daemon Is Accessible

If you have started the **pcnfsd** daemon at the command line, you should verify that the daemon is accessible through the network. Use the **rpcinfo** command at the command line. After you have entered the command, you should receive the message that the program is ready and waiting.

Related Information

Network File System (NFS) Overview for System Management on page 11–1.
How to Configure NFS on page 11–63, How to Configure NFS Servers and Clients on page 11–67, How to Configure an NFS Server on page 11–66, How to Configure an NFS Client on page 11–65, How to Configure PC-NFS on page 11–69, How to Configure the inetd Daemon on page 14–102.

Maintaining NFS

Once NFS has been configured on the system, changes can be made to the configuration without going through the entire NFS installation procedure. This article discusses maintenance tasks, such as adding network daemons, exporting and unexporting directories, and mounting and unmounting file systems.

Unless you are administering a non-networked stand-alone machine, you need an understanding of the activities involved in maintaining an NFS server and maintaining an NFS client.

Related Information

Network File System (NFS) Overview for System Management on page 11–1.

Configuring NFS on Your System on page 11–25, Maintaining an NFS Server on page 11–31, Maintaining an NFS Client on page 11–34, Troubleshooting NFS-Related Problems on page 11–37, Administering Secure NFS on page 11–11.

Functional List of NFS and NIS Daemons and Commands on page 11–110.

Maintaining an NFS Server

The primary method of maintaining an NFS Server is by making changes to the current configuration of the system. This may include changing the number of **nfsd** daemons that are active on the system or the configuration of the **inetd** configuration database. The most common task is maintaining the list of exported directories on a particular server.

After an NFS server is configured, only the basic NFS service has been established.

Specific tasks that may need to be performed after the initial configuration is complete include the following:

- Changing the number of **nfsd** daemons that are active on the system
- Modifying the **inetd** configuration database to reflect needed changes
- Exporting or unexporting directories on the server or modifying the **/etc/exports** file
- Exporting and unexporting directories directly during a server's work session using the **exportfs** command
- Enabling root user access over the network.

Some additional NFS-related activities that you need to consider when maintaining an NFS server include:

- Troubleshooting and fixing NFS-related problems as they arise
- Administering secure NFS networking for your network as determined by your site's requirements.

Changing the Number of Network Daemons

The number of active **nfsd** daemons running on an NFS server may be changed by using the **chnfs** command. Because the NFS daemons are controlled by the System Resource Controller (SRC), you need to change the SRC database to reflect the number of **nfsd** daemons running on a system. The **chnfs** command automatically makes these changes for you. Use the **chssys** command to change the parameters passed to any SRC controlled daemon, including the **nfsd** daemon. The **chnfs** command stops the currently running **nfsd**

daemons, makes the change in number, and then starts the new number of daemons on the system.

Modifying the inetd Configuration Database

The **inetd** configuration database is represented by the **/etc/inetd.conf** file. Changes made to this file are not reflected until the **inetd** configuration database is rebuilt. To rebuild the **inetd** configuration database, you need to modify the **/etc/inetd.conf** file and use the **inetimp** command. Once the database is rebuilt, you need to refresh the **inetd** daemon. Full details of this procedure are explained in How to Configure the inetd Daemon on page 14–102.

Exporting and Unexporting Directories

An NFS server has **nfsd** daemons and the **rpc.mountd** daemon running on the system and has directories exported using the **exportfs** command. Clients of this server are only restricted by the specifications made in the **/etc/exports** file. The **/etc/rc.nfs** file executes the **exportfs** command to export directories at system restart. It checks for the presence of the **/etc/exports** file and then issues the **exportfs –a** command, which exports all entries in the file.

The **/etc/exports** file entries may be updated, added, or deleted in three different ways. The SMIT command may be used with a fastpath parameter to reach the menus that deal with NFS exports. The fastpath IDs are as follows:

```
smit mknfsexp
smit chnfsexp
smit rmnfsexp
```

These menus create an NFS export, change an NFS export, and remove an NFS export, respectively.

Corresponding to each of the SMIT fastpath methods is the underlying command that is executed. The three commands would be:

```
mknfsexp
chnfsexp
rmnfsexp
```

The **/etc/exports** file may be edited directly to add, change, or remove entries. After the **/etc/exports** file is edited, the **exportfs** command should be executed.

Exporting Directories Using the exportfs Command

The **exportfs** command may be used to export a directory temporarily. An entry does not have to be added to the **/etc/exports** file to export a directory. If the **exportfs –i** command is executed, the **/etc/exports** file is not checked for the specified directory and all options are taken directly from the command line.

The **exportfs** command is particularly useful for exporting directories that you want to make available for only a specific amount of time before they are unexported.

Whenever the **exportfs** command is executed, it updates the **/etc/xtab** file, which lists currently exported directories. Although the **/etc/xtab** file has a format identical to the **/etc/exports** file, the **/etc/xtab** file changes whenever the **exportfs** command is executed. Conversely, the **/etc/exports** file must be changed manually by the user.

For example, when explicitly specifying a directory with the **exportfs** command, that directory is automatically listed in the **/etc/xtab** file but not in the **/etc/exports** file. If the

directory is unexported with the **exportfs –u** command, the unexported directory is no longer listed in the **/etc/xtab** file.

Note: The **/etc/xtab** file is updated automatically, and should not be edited.

Enabling Root User Access Over the Network

When you export a file system you can enable the root user on a particular machine to have root access to that file system by modifying the contents of the **/etc/exports** file. By default, the root user is not granted root access on remotely mounted file systems. When a person logged in as the root user on one host requests access to a particular file from NFS, the user ID of the requester is changed to the user ID of user name `nobody`. (`nobody` is one of the user names automatically placed in the default **/etc/passwd** file.) The access rights of user `nobody` are the same as those given to the public, or to *others* for a particular file. For example, if the public only has execute permission for a file, then user `nobody` can only execute that file.

When you export a file system, you can permit the root user on a particular machine to have root access to that file system by editing the **/etc/exports** file on the server. You can enable root user access for one client, several clients, or all clients by using the **/etc/exports** file.

Related Information

The **chnfs** command, **chssys** command, **chnfsexp** command, **exportfs** command, **mknfsexp** command, **rmnfsexp** command, **smit** command.

The **biod** daemon, **inetd** daemon, **mountd** daemon.

The **/etc/exports** file, **/etc/xtab** file, **/etc/inetd.conf** file.

Network File System (NFS) Overview for System Management on page 11–1.

How to Configure the inetd Daemon on page 14–102, How to Export a Directory Using NFS on page 11–79, How to Unexport a Directory Using NFS on page 11–86, How to Use the NFS Secure Option When Exporting a Directory on page 11–78.

Troubleshooting NFS-Related Problems on page 11–37.

Maintaining an NFS Client

This article contains information related to setting up and maintaining an NFS client on a network. This information is useful if you are responsible for administering your client machine or if you are administering both NFS servers and NFS clients. The maintenance activities for administering a client include changing the number of active **biod** daemons, making mount points, and mounting and unmounting directories from an NFS server.

Changing the Number of biod Daemons

The number of active **biod** daemons running on an NFS server may be changed by using the **chnfs** command. Because the NFS daemons are controlled by the System Resource Controller (SRC), you need to change to the SRC database to reflect the number of **biod** daemons running on a system. The **chnfs** command automatically makes these changes for you. The **chnfs** command stops the currently running **biod** daemons, makes the change in number, and then starts the new number of daemons on the system.

Making Mount Points

NFS provides the ability to access directories and files that reside on a remote system through the NFS mount process. The locations within a directory tree through which a client accesses remote directories are called mount points. Mount points are empty directories that can be created using the **mkdir** command.

Mounting and Unmounting Directories

NFS mounting can occur during system restart when the **/etc/rc.nfs** file is executed. The information for these predefined mounts comes from the **/etc/filesystems** file. Mounting and unmounting of directories can also be done explicitly by using the **mount** and **umount** commands. Mounts can also be done as needed by using the **automount** daemon to handle automatic mounts.

Choosing a Mount Method

The methods for mounting and unmounting directories are *predefined*, *explicitly defined*, and *automatic* mounts.

- Predefined mounts are achieved by adding the appropriate entry to the **/etc/filesystems** file.
- Explicit mounts are achieved by using the mount and umount commands.
- Automatic mounts are achieved by defining automount maps and using the automount daemon.

The information for predefined mounts is stored in the **/etc/filesystems** file in the form of stanzas. These stanzas or entries in the **/etc/filesystems** file should be used when mounts are always required for proper operation of a client.

Explicit mounts serve the needs of the root user. They should be used when there is a requirement for occasional unplanned mounts. Explicit mounts can also be used if a mount is required for special tasks and that mount should not be generally available on the NFS client.

Automatic mounts are controlled by the **automount** daemon. Be aware that each time the **automount** daemon is executed, another daemon is started on the system. To keep the total number of daemons low, the preferred method for using the **automount** daemon is to use the predefined special maps.

Establishing the Predefined NFS Mounts

Predefined NFS mounts refer to the remote directories that are mounted upon system restart. The method for configuring the predefined mounts involves modifying the **/etc/filesystems** file and adding the correct mount command to the **/etc/rc.nfs** file.

The modification of the **/etc/filesystems** file can be done either by editing the file or by using the **mknfsmnt** command to add an NFS mount stanza. For each NFS mount that is to be predefined a corresponding entry or stanza needs to be added to the **/etc/filesystems** file. For a description of the **/etc/filesystems** file and available options, see How to Edit the /etc/filesystems File on page 11–88 and the description of the **/etc/filesystems** file.

After the stanzas have been added to the **/etc/filesystems** file, a mount command may be added to the **/etc/rc.nfs** file to actually execute those mounts on system restart. The command to be added is as follows:

```
mount –v nfs
```

This command attempts all of the mounts that have the following two options set in the stanzas:

```
vfs = nfs
mount = true
```

Note: The `mount = true` statement has to be present in order for the **mount** command to work. Also note that the stanzas may also be used for explicit mounts. If the **mount** command is used with a parameter of a mount point that is present as a stanza in the **/etc/filesystems** file, then the **mount** command uses the options listed in the stanza to attempt the mount.

Using the mount and umount Commands

The **mount** and **umount** commands explicitly mount and unmount a file system. These commands are particularly useful for directories that are to be accessed for only a short amount of time. These commands require that the server be accessible over the network and be exporting the directory. The **mount** command explicitly requires that the user have root authority.

Directories accessed through the **mount** command remain mounted during a work session unless explicitly unmounted with the **umount** command. If the system is restarted, the mount will no longer be in effect unless a stanza was added to the **/etc/filesystems** file to make the mount a predefined one.

Using the **mount** command explicitly requires that the user provide all the information on the command line to mount the file system. At a minimum the user must provide the server's host name, the path name of the remote directory, and the path name of the local directory that is to be used as the mount point. All other options used by NFS can be specified on the command line.

Note: It is possible to use the **mount** command and only supply the mount point as a parameter. As mentioned earlier, the **mount** command uses the mount point to locate the corresponding stanza in **/etc/filesystems** file. This stanza is then used to supply the needed information to complete the mount.

Using the automount Daemon

The **automount** daemon automatically and transparently mounts a file system from NFS servers whenever a file or directory within that file system is opened. The **automount** daemon appears to the kernel to be an NFS server. Lookups on the specified directory are intercepted by this daemon, which attempts to mount the file system and respond to the request. Information about which server to mount from, the remote directory path, and the appropriate mount options is gathered from the maps specified when the **automount** daemon is started.

Once mounted, files in the specified directory are made available using a symbolic link to the real mount point with a temporary directory. If the specified directory does not exist, the daemon creates it and then removes it automatically when the daemon exits.

Since the name-to-location binding is dynamic within the **automount** daemon, updates to a network information service map used by the **automount** daemon are transparent to the user. Because of this, there is no need to pre-mount shared file systems for applications that have hard-coded references to files and directories. Nor is there a need to maintain records of which hosts must be mounted for what applications.

For detailed information on using the **automount** daemon, see How to Mount a Filesystem using NFS on page 11–80.

Related Information

The **mkdir** command, **mount** command, **umount** command, **smit** command.

The **automount** daemon, **biod** daemon.

Maintaining an NFS Server on page 11–31.

Troubleshooting NFS–Related Problems

As with other network services, problems can occur on machines that use NFS. Troubleshooting for these problems involves understanding the strategies for tracking NFS problems, recognizing NFS-related error messages, and selecting the appropriate solutions.

The strategies for detecting NFS problems involve the following:

- Determining where NFS service became inoperable
- Identifying hard-mounted and soft-mounted file problems
- Identifying server problems
- Identifying client problems
- Understanding error messages for mount problems
- Fixing hung programs
- Identifying the cause of slow access times
- Understanding permissions and authentication schemes
- Understanding name resolution on an NFS server
- Mounting from servers that have earlier versions of NFS
- Understanding problems that may occur if the NFS kernel extension is not loaded.

When solving NFS-related problems, it is helpful to recognize the names and functions of the following commands and daemons:

mountd	Answers requests from clients for file system mounts.
nfsstat	Displays information about a machine's ability to receive calls.
on	Executes commands on remote machines.
portmap	Maps RPC program numbers to port numbers. When a client makes a remote procedure call to a specific program, it contacts the **portmap** daemon to determine the port where the request should be sent.
rpcinfo	Reports the status of RPC servers.
rstatd	Returns performance statistics obtained from the kernel.
rup	Shows the status of a remote host on the local network.
rusers	Reports a list of users logged on to the remote machines.
rusersd	Responds to queries from the **rusers** command.
rwall	Sends messages to all users on the network.
rwalld	Handles requests from the **rwall** command.
showmount	Displays a list of all clients that have mounted remote file systems.
spray	Sends a specified number of packets to a host.
sprayd	Receives packets sent by the **spray** command.

Determining Where NFS Service Became Inoperable

When tracking down an NFS problem, isolate each of the three main points of failure to determine which is not working: the server, the client, or the network itself.

The following example illustrates a mount request and summarizes how the mount works and where it can become inoperable.

```
mount –n dancer /usr/src /dancer.src
```

This command requests server `dancer` to send a file handle (*fhandle*) for the `/usr/src` directory. This file handle is sent to the client kernel by the `mount` program.

The client kernel looks up the `/dancer.src` directory. If there are no problems, the command ties the file handle to the directory in a mount record. All future file system requests to that directory and its subdirectories go through the file handle to server `dancer`.

The **mountd** daemon must be present on the NFS server for a remote mount to succeed. Make sure the **mountd** daemon is running on the server by issuing the following command:

```
ps –ef | fgrep mount
```

You should get a response that contains a line similar to one of the following:

```
root   5552   1 0    Feb 23     –   0:47 /usr/etc/rpc.mountd
root   5552   1 0    Feb 23     –   0:47 rpc.mountd
```

This indicates that the `rpc.mountd` daemon is running on the server. If the `rpc.mountd` daemon does not show up in the daemon list, it must be started. Modify the **/etc/rc.nfs** file so that the **mountd** daemon will be started.

You can also start the **mountd** daemon from the command line if you are the root user by entering:

```
startsrc –s rpc.mountd
```

Remote mount also needs **nfsd** daemons (8 are typically started) to execute on NFS servers. Make sure the **nfsd** daemons are running on the server by issuing the following command:

```
ps –ef | fgrep nfsd
```

You should see a line of output for each **nfsd** daemon running. If none are running, check the **/etc/rc.nfs** file for the following lines:

```
if [ –x /usr/etc/nfsd –a –f /etc/exports ]; then
> /etc/xtab
/usr/etc/exportfs –a
startsrc –s nfsd
```

If the NFS server's **/etc/rc.nfs** file does not enable **nfsd** daemons, add these lines, or your own version of them, to the file. You can enable these daemons without rebooting. To do so, log in as the root user and enter the following at the command line:

```
startsrc –s nfsd
```

Recognizing Debugging Hints

NFS may become inoperable due to problems at the server, the client, or the network itself. Some areas of concern for debugging NFS are:

- Identifying hard-mounted and soft-mounted file problems
- Identifying server problems
- Identifying client problems
- Identifying the network connection.

Identifying Hard-Mounted and Soft-Mounted File Problems

When the network or server has problems, programs that access hard-mounted remote files fail differently from those that access soft-mounted remote files. Hard-mounted remote file systems cause programs to retry until the server responds again. Soft-mounted remote file systems return an error after trying unsuccessfully for a while.

If the server for a remote file system fails to respond, it retries the mount request until it succeeds. When you use the **mount** command with the **bg** option, it retries the mount in the background.

Once a hard mount succeeds, programs that access hard-mounted files hang as long as the server fails to respond. In this case, NFS prints the following message:

```
NFS server not responding
```

On a soft-mounted file system, programs get the following message when they access a file on a server that is dead.

```
Connection timed out
```

Unfortunately, many programs do not check return conditions on file system operations, so you may not see this error message when accessing soft-mounted files. However, an NFS error message should print on the console in this case.

Identifying Server Problems

If a client is having NFS trouble:

- Verify that the server is up and running.
- Verify that the mountd daemon is running.
- Verify that the network connections are good.
- Verify that the client daemons are running.

Verify That the Server Is Up and Running

Enter the following at the command line of a client workstation:

```
/usr/etc/rpcinfo -p server_name
```

If the server is up, a list of programs, versions, protocols, and port numbers is printed, similar to the following:

```
program  vers proto   port
100000    2   tcp    111  portmapper
100000    2   udp    111  portmapper
100005    1   udp   1025  mountd
100001    1   udp   1030  rstatd
100001    2   udp   1030  rstatd
100001    3   udp   1030  rstatd
100002    1   udp   1036  rusersd
100002    2   udp   1036  rusersd
100008    1   udp   1040  walld
100012    1   udp   1043  sprayd
100005    1   tcp    694  mountd
100003    2   udp   2049  nfs
100024    1   udp    713  status
100024    1   tcp    715  status
100021    1   tcp    716  nlockmgr
100021    1   udp    718  nlockmgr
100021    3   tcp    721  nlockmgr
100021    3   udp    723  nlockmgr
100020    1   udp    726  llockmgr
100020    1   tcp    728  llockmgr
100021    2   tcp    731  nlockmgr
```

Verify That the mountd Daemon Is Running

If the server is up, use the **rpcinfo** command to verify that the **mountd** daemon is running:

```
/usr/etc/rpcinfo –u server_name mount
```

The program number (`100005`) corresponds to the `mountd` command in the previous file list. If the daemon is running at the server, the following response is returned:

```
program 100005 version 1 ready and waiting
```

If a similar response is not returned, try logging in at the server console to check its operation.

Verify That the Network Connections Are Good

If the server is up and the **mountd** daemon is running, but your client machine cannot communicate with it, check the network connections between your machine and the server.

Verify That the Client Daemons Are Running

If the server and network connections are good, use the **ps** command to check your client daemons. A **portmap** daemon and several **biod** daemons should be running.

Identifying Client Problems

The following activities can help track the source of mounting problems from a client workstation. If problems arise when you attempt to mount a file system, verify the following:

- A valid mount point exists for the file system being mounted.
- The name of the file system exists in the **exports** file on the server, and it is exported properly by the server.
- The server is up and running. Enter the following at the command line to check the server:

  ```
  /usr/etc/rpcinfo -p server_name
  ```

  Replace the *server_name* parameter with the name of the server you are checking. If the server is up and running, a list of available programs should print on the console.
- For remote mounts, the number of **nfsd** daemons to execute on NFS servers is specified in the **/etc/rc.nfs** file, and the **nfsd** daemons are running.
- For remote mounts, the **mountd** daemon exists in the **/etc/rc.nfs** file and is running. Enter the following at the command line to check the **mountd** daemon:

  ```
  /usr/etc/rpcinfo -u server_name mountd
  ```

  Replace the *server_name* parameter with the name of the server whose file system you are trying to mount. If the **mountd** daemon is running, a ready-and-waiting response appears on your console.
- The client daemons are running. Check to see that the **portmap** and **biod** daemons exist and are running.
- The network connection is working properly.

Understanding Error Messages for Mount Problems

A remote mounting process can fail in several ways. The error messages associated with mounting failures are as follows:

- `mount: ... already mounted`

  Here, the file system that you are trying to mount is already mounted.
- `mount: ... not found in /etc/filesystems`

  Here, the specified file system or directory name cannot be matched. If you issue the **mount** command with either a directory or file system name but not both, the command looks in the **/etc/filesystems** file for an entry whose file system or directory field matches the argument. If the **mount** command finds an entry such as the following:

  ```
  /dancer.src:
        dev=/usr/sr
        nodename = d61server
        type = nfs
        mount = false
  ```

  then it performs the mount as if you had entered the following at the command line:

```
/usr/etc/mount -n dancer -o rw,hard /usr/src /dancer.src
```

- `... not in hosts database`

  On a network without NIS, this message indicates that the host specified to the **mount** command is not in the **/etc/hosts** file. On a network running NIS, the message indicates that NIS could not find the host name in the **/etc/hosts** database or that the NIS **ypbind** daemon on your machine has died. If the **/etc/resolv.conf** file exists so that the name server is being used for host name resolution, there may be a problem in the **named** database. See Understanding Name Resolution on an NFS Server on page 11–45.

  Check the spelling and the syntax in your **mount** command. If the command is correct, your network does not run NIS, and you only get this message for this host name, check the entry in the **/etc/hosts** file.

  If your network is running NIS, make sure that the **ypbind** daemon is running by entering the following at the command line:

```
ps –ef
```

  You should see the **ypbind** daemon in the list. Try using the **rlogin** command to log in remotely to another machine, or use **rcp** to remote-copy something to another machine. If this also fails, your **ypbind** daemon is probably dead or hung.

  If you get only the following message for this host name, you should check the **/etc/hosts** entry on the NIS server.

- `mount: ... server not responding: port mapper failure – RPC timed out`

  Either the server you are trying to mount from is down, or its port mapper is dead or hung. Try rebooting the server to restart the **inetd**, **portmap**, and **ypbind** daemons.

  If you cannot log in to the server remotely with the **rlogin** command but the server is up, you should check the network connection by trying to log in remotely to some other machine. You should also check the server's network connection.

- `mount: ... server not responding: program not registered`

  This means that the **mount** command got through to the port mapper, but the **rpc.mountd** NFS mount daemon was not registered.

- `mount: access denied ...`

  Your machine name is not in the export list for the file system you are trying to mount from the server.

  You can get a list of the server's exported file systems by running the following command at the command line:

```
showmount –e hostname
```

  If the file system you want is not in the list, or your machine name or netgroup name is not in the user list for the file system, log in to the server and check the **/etc/exports** file for the correct file system entry. A file system name that appears in the **/etc/exports** file, but not in the output from the **showmount** function, indicates a failure in the **mountd** daemon. Either the daemon could not parse that line in the file, it could not find the directory, or the directory name was not a locally mounted directory. See the **/etc/exports** documentation for more information.

  If the **/etc/exports** file looks correct and your network runs NIS, check the server's **ypbind** daemon. It may be dead or hung.

- `mount: ...: Permission denied`

  This message is a generic indication that some part of authentication failed on the server. It may be that in the previous example, you are not in the export list, the server could not recognize your machine's **ypbind** daemon, or that the server does not believe you are who you say you are.

  Check the server's **/etc/exports** file, and, if applicable, the **ypbind** daemon. In this case you can just change your host name with the **hostname** command and retry the **mount** command.

- `mount: ...: Not a directory`

  Either the remote path or the local path is not a directory. Check the spelling in your command and try to run on both directories.

- `mount: ...: You are not allowed`

  You must have root authority or be a member of the system group to run the **mount** command on your machine because it affects the file system for all users on that machine. NFS mounts and unmounts are only allowed for root users and members of the system group.

Fixing Hung Programs

If programs hang during file-related work, the NFS server may be dead. In this case, the following error message may be displayed:

```
NFS server hostname not responding, still trying
```

The NFS server (`hostname`) is down. This indicates a problem with the NFS server, the network connection, or the NIS server.

If your machine hangs completely, check the servers from which you have mounted file systems. If one or more of them is down, do not be concerned. When the server comes back up, your programs continue automatically. No files are destroyed.

If a soft-mounted server dies, other work is not affected. Programs that time out trying to access soft-mounted remote files fail with the `errno` message, but you will still be able to access your other file systems.

If all servers are running, determine whether others who are using the same servers are having trouble. More than one machine having service problems indicates a problem with the server's **nfsd** daemons. In this case, log in to the server and run the **ps** command to see if the **nfsd** daemon is running and accumulating CPU time. If not, you may be able to kill and then restart the **nfsd** daemon. If this does not work, reboot the server.

If other systems seem to be up and running, check your network connection and the connection of the server.

Identifying the Cause of Slow Access Times for NFS

If access to remote files seems unusually slow, ensure that access time is not being inhibited by a runaway daemon, a bad **tty** line, or a similar problem. At the server, enter the following at the command line:

```
ps –ef
```

If the server seems fine and other users are getting timely responses, make sure your **biod** daemons are running. Try the following steps:

1. Run the **ps –ef** command and look for the **biod** daemons in the display.

   If they are not running or are hung, continue with these steps:

2. Stop the **biod** daemons that are in use by issuing the following command:

   ```
   stopsrc –x biod –c
   ```

3. Start the **ciod** daemons by issuing the following command:

   ```
   startsrc –s biod
   ```

   To determine if the **biod** daemons are hung, run the **ps** command as above, copy a large file from a remote system, and then run the **ps** command again. If the **biod** daemons do not accumulate CPU time, they are probably hung.

   If the **biod** daemons are working, check the network connection. The **netstat –i** command determines whether you are dropping packets. Use the **nfsstat –c** and **nfsstat –s** commands to determine if the client or server is retransmitting large blocks. A retransmission rate of 5% is considered high.

Understanding Permissions and Authentication Schemes

Sometimes, after mounts have been successfully established, there are problems in reading, writing, or creating remote files or directories. Such difficulties are usually due to permissions or authentication problems. Permission and authentication problems can vary in cause depending on whether NIS is being used and secure mounts are specified.

The simplest case occurs when nonsecure mounts are specified and NIS is not used. In this case, user and group IDs are mapped solely through the server and clients **/etc/passwd** and **/etc/group** files, respectively. In this scheme, for a user named `john` to be identified both on the client and on the server as `john`, the user `john` in the **/etc/passwd** file must have the same UID number. The following is an example of how this might cause problems:

```
User john is uid 200 on client foo.
User john is uid 250 on server bar.
User jane is uid 200 on server bar.
```

The `/u/bar` directory is mounted from server `bar` onto client `foo`. If user `john` is editing files on the `/u/bar` remote file system on client `foo`, confusion results when he saves files.

The server `bar` thinks the files belong to user `jane`, because `jane` is UID `200` on `bar`. If `john` logs on directly to `bar` by using the **rlogin** command, he may not be able to access the files he just created while working on the remotely mounted file system. `jane`, however, is able to do so because the machines arbitrate permissions by UID, not by name.

The only permanent solution to this is to re-assign consistent UIDs on the two machines. For example, give `john` UID 200 on server `bar` or 250 on client `foo`. The files owned by `john` would then need to have the **chown** command run against them to make them match the new ID on the appropriate machine.

Because of the problems with maintaining consistent UID and GID mappings on all machines in a network, NIS is often used to perform the appropriate mappings so that this type of problem is avoided. NIS maintains a database that takes care of the mappings of UID and GID identities across the network. See How to Configure NIS on page 11–71 for more information.

Understanding Name Resolution on an NFS Server

When an NFS server services a mount request, it looks up the name of the client making the request. The server takes the client IP address and looks up the corresponding host name that matches that address. Once the host name has been found, the server looks at the exports list for the requested directory and checks the existence of the client's name in the access list for the directory. If an entry exists for the client and the entry matches exactly what was returned for the name resolution, then that part of the mount authentication passes.

If the server is not able to do the IP address-to-host-name resolution, the server denies the mount request. The server must be able to find some match for the client IP address making the mount request. If the directory is exported with the access being to all clients, the server still must be able to do the reverse name lookup to allow the mount request.

The server also must be able to look up the correct name for the client; for example, if there exists an entry in the **/etc/exports** file like the following:

```
/tmp    –access=silly:funny
```

and these corresponding entries exist in the **/etc/hosts** file:

```
150.102.23.21     silly.domain.name.com
150.102.23.52     funny.domain.name.com
```

Notice that the names do not correspond exactly. When the server looks up the IP address-to-host-name matches for the hosts `silly` and `funny`, the string names do not match exactly with the entries in the access list of the export. This type of name resolution problem usually occurs when using the **named** daemon for name resolution. Most **named** daemon databases have aliases for the full domain names of hosts so that users do not have to enter full names when referring to hosts. Even though these host-name-to-IP address entries exist for the aliases, the reverse lookup may not exist. The database for reverse name lookup (IP address to host name) usually has entries containing the IP address and the full domain name (not the alias) of that host. Sometimes the export entries are created with the shorter alias name, causing problems when clients try to mount.

Mounting from NFS Servers That Have Earlier Version of NFS

When mounting an AIX Version 3 client to a server that has an earlier version of NFS, a problem occurs when the user on the client executing the mount is a member of more than eight groups. Some servers are not able to deal correctly with this situation and deny the request for the mount. The solution is to change the user's group membership to a number less than eight and then retry the mount. The following error message is characteristic of this group problem:

```
RPC:  Authentication error; why=Invalid client credential
```

Understanding Problems That Occur if the NFS Kernel Extension Is Not Loaded

Some NFS commands do not execute correctly if the NFS kernel extension is not loaded. Some commands with this dependency are: **nfsstat, exportfs, mountd, nfsd,** and **biod.** When NFS is installed on the system, the kernel extension is placed in the **/etc/nfs.ext** file. This file is then loaded as the NFS kernel extension when the system is configured. The script that does this kernel extension loads the **/etc/rc.net** file. There are many other things done in this script, one of which is to load the NFS kernel extension. It is important to note that TCP/IP kernel extension should be loaded before the NFS kernel extension is loaded.

Note: The NFS kernel extension should not be loaded more than once during one system restart. If it is loaded more than once, the results are unpredictable.

Related Information

The **hostname** command, **mount** command, **rcp** command, **rlogin** command, **showmount** command.

The **inetd** daemon, **named** daemon, **mountd** daemon, **nfsd** daemon, **portmap** daemon, **ypbind** daemon.

The **/etc/inetd.conf** file.

Alphabetical List of NFS Daemons and Commands on page 11–107, Functional List of NFS Daemons and Commands on page 11–110.

Maintaining NFS on page 11–31.

Network File System (NFS) Overview for System Management on page 11–1.

Network Information Service (NIS) Overview for System Management

The IBM NFS network information service (NIS) is a network service used to distribute system information on networked hosts. An NIS environment is composed of NIS *clients* and NIS *servers* logically grouped together in a domain using *maps,* or databases, that provide system information such as passwords and host names.

All of the NIS maps for a domain are stored in the **/etc/yp** directory. They are in a directory named after the domain.

The host that maintains the NIS maps for a domain is called the NIS *master server.* To balance the NIS processing load and provide services when the NIS master server is unavailable, additional hosts can be designated as NIS *slave servers.* The NIS slave servers maintain exact replicas of the NIS master server maps. You must configure the NIS master server before configuring any slave servers. Use at least one NIS slave server for each domain. The remaining hosts should be designated as NIS *clients.*

You must configure the master server, slave servers, and client machines, before you can modify them. The master server maintains the NIS maps and must be configured first. Slave servers maintain exact replicas of the NIS master server maps. Remaining hosts should be designated as NIS clients.

NIS clients do not maintain any NIS maps. They query the NIS servers for system and user account information. The NIS clients do not make a distinction between querying the NIS master server or an NIS slave server.

Changes to the NIS maps are made on the NIS master server and then *propagated* (transferred) or pushed to the NIS slave servers.

Note: The NIS service algorithm requires that changes to the database be made only on the NIS master server.

A group of hosts that share an NIS map belong to the same domain. The hosts are usually grouped together in the domain for a common reason; for example, when working in the same group at a particular location. Each NIS host is assigned to a domain when the system starts. The person who manages your system (or a user with root-user authority) can assign a host to a different domain or change the name of the domain with the **chypdom** command. The domain name must be set on all hosts that intend to use NIS.

Access to data served by NIS is independent of the relative locations of an NIS client and server. To access system information maintained in an NIS map, an NIS client makes a remote procedure call (RPC) to an NIS server. The NIS server searches its local database and returns the requested information to the NIS client.

Related Information

The **chypdom** command.

Alphabetical List of NFS and NIS Daemons and Commands on page 11–107, Functional List of NFS and NIS Daemons and Commands on page 11–110.

List of NIS Concepts, List of NIS Procedures.

Network File System (NFS) Overview for System Management on page 11–1.

Understanding NIS Maps on page 11–51, Understanding NIS Servers on page 11–53, Maintaining NIS on page 11–54.

Understanding NIS Services

NIS is a distributed database system to share system information. NIS is used when there are a number of systems that need to share system information. This information includes the **/etc/passwd** file, the **/etc/group** file, and the **/etc/hosts** file. NIS reduces the effort to maintain a large number of systems and provides a global user ID and group ID name space.

Without NIS, having a large number of systems on the network can create problems for the user. The user would need to remember a different password for every system they use, or else maintain the same password for every system themselves, and update their password on each system separately.

NFS can be used to make the user's directories and files available on every system. Since NFS does not try to translate user IDs or group IDs between systems, the person administering the system must ensure that each user in the network has the same user ID and group ID on every system. If this is not done, a user on one system who creates a file and logs into another system may find that the file is owned by another user.

Using an automated system like NIS makes system maintenance easier and less time-consuming, and removes the need for users to do their own password maintenance.

Maintaining Consistent System Information without NIS

NIS is only one way to automate this process of maintaining global systems information. Another way is for the person administering the system to automate the process themselves. One system is usually chosen as the authoritative source for files like **/etc/passwd**, **/etc/group** and other associated files. The updates that need to be done are executed on this one system and then this information is copied to all systems that are within a predefined set.

There are some drawbacks to this approach. If one of the systems is unreachable during the copying phase, that system has to be updated at a later time. During the period of time that the system has not been updated it contains inconsistent information. This inconsistency may cause a security problem or just an inconvenience for the users involved. Another drawback to this approach is that the amount of information being copied over the network becomes large for a large number of systems. The entire contents of all of the involved files must be copied on every update to every system. The longer the time it takes to copy the data, the greater the potential problem. Also, not all of the required system information may be copied correctly, in which case the system may not work properly.

Maintaining Consistent System Information with NIS

The information in this section is designed to assist you in maintaining consistent system information with NIS.

Choosing a Master Server

With NIS, as with other ways of maintaining consistent system information, one system should be chosen to contain all of the system information for a set of hosts. Some of the characteristics of this NIS server should be the following:

- Accessible by the person who administers the system. If something goes wrong, or if updates need to be made, it should be easy to reach the master server.

- Stable. The master server is usually a system that stays active for long periods of time. It needs to be stable so that the systems that depend on this master server enjoy uninterrupted service.
- Accessible from the network. Some networks become complex with the presence of many gateways or bridges. The master server should be highly accessible from the majority of systems on the network.

Identifying Clients

Once the master server has been chosen, the other systems within the predefined set of systems use it to retrieve the needed system information. These other systems are called clients. When a client needs a piece of system information, such as a entry from the **/etc/passwd** file, it makes a request to the master server. If the information exists, the master server responds with the information. The client system then continues as usual. This way the client system only asks for the information that is needed. The system information is kept consistent because there is only one copy of it among all of the systems. There is not a large amount of data copied over the network because the clients only ask for the information that they require.

Understanding the NIS Domain

An NIS domain is a collection of systems that are logically grouped. This group of systems are usually contained within one administrative domain. The systems are maintained by one entity and the users that use the systems are members of some group themselves. The NIS domain is defined as follows:

- There is one master server per NIS domain.
- Systems within the NIS domain use the master server to retrieve system information.
- Systems within the NIS domain, including the master server, set the domain name with the **domainname** command.
- Clients modify system information files to use NIS.

The systems in the domain are typically on the same network. The number of systems that are contained in a domain is moderate. As the number of systems grows the response time from the master server increases because of the increased work load. This server load can be reduced by defining more than one domain or by adding slave servers.

Adding Slave Servers

Having one system as the master server for an NIS domain can be a problem for two reasons. First, if the master server fails or the network fails and as a result the master server becomes unreachable, the clients in the domain are going to be held because they cannot retrieve the needed information. Second, when the number of clients grows or the load that the clients place on the master server increases the response time becomes very slow.

The way to solve these problems is to add one or more extra servers to the NIS domain. By design, another master server cannot be added because there would be two authoritative sources. There is another type of server called a *slave server* that can help with the extra load and failure cases without violating the authority of the master server. The slave server is added to the domain and is able to answer the same queries that the master is able to answer. To achieve this, the master server copies all updates to its database information to all of the slave servers in the master server's domain.

This process does not share the disadvantages of copying all data to all systems, because the amount of data copied and the number of slave servers should still be relatively small. The number of slave servers in a domain should be balanced to achieve the desired level of

availability and response time without adding the expense of copying data to too many systems.

NIS Service Time

Wait time for NIS may increase if all servers within a domain are unreachable. If the client makes a request for information and the servers are unreachable, the client waits until a server responds to its request. This waiting time varies depending on the reason the servers are unreachable. Adding slave servers increases the availability of information.

Every request for system information requires a server contact. The speed of your network can affect the response time. A local retrieve is usually faster than a network retrieve. This time is a compromise between ease of use and global user and group ID name space.

Related Information

Network File System (NFS) Overview for System Management on page 11–1.

Network Information Service (NIS) Overview for System Management on page 11–47 .

Understanding NIS Maps on page 11–51, Understanding NIS Servers on page 11–53, Maintaining NIS on page 11–54.

Understanding NIS Maps

NIS Maps are constructed from standardized text files. The default NIS maps are created from the following files if they are available on the master server:

- **/etc/ethers**
- **/etc/netgroup**
- **/etc/networks**
- **/etc/aliases**
- **/etc/passwd** (or a specially designated NIS password account file)
- **/etc/group** (or a specially designated NIS group account file)
- **/etc/hosts**
- **/etc/rpc**
- **/etc/protocols**
- **/etc/netmasks**
- **/etc/bootparams**

The information in an NIS map is in DBM/NDBM format. These maps are located in the subdirectories of the **/etc/yp** directory on NIS servers. The name of this subdirectory is the name of the NIS domain. For example, maps for the `literature` domain are located in the **/etc/yp/literature** subdirectory on NIS servers.

The most commonly used maps have *nicknames*. Some commands translate map names into map nicknames. For instance, when you enter `ypcat hosts`, it is translated into `ypcat hosts.byname`, since there is no file called `hosts` in the NIS database. The **ypcat –x** command furnishes a list of expanded nicknames.

There are two kinds of NIS maps, standard and nonstandard. Standard NIS maps are those that come from NIS installation. Nonstandard maps are created for a specific application.

A map can be created by creating a text file that can be converted into a map by the **makedbm** command. Rules can be added to the **/etc/yp/Makefile** file to make the new map when the standard NIS maps are made. A map can also be made by entering input to the **makedbm** command through standard input. For more information on these methods, see How to Create a New NIS Map from a Text File on page 11–98 and How to Create a New NIS Map Without a Text File on page 11–99.

Database information is maintained in NIS *maps*. Each NIS map is created by associating an index *key* with a *value*. For example, the information in the NIS master server's **/etc/hosts** file is used to create an NIS map that uses each host name as a key. The value key and value pairs (also known as *records*) that are created from the entries in the **/etc/hosts** file comprise the NIS host-name map.

The NIS maps are created using the **makedbm** command, which converts input into DBM format files. An NIS map consists of two DBM files: **map.key.pag** and **map.key.dir**. For example, the host name map in NFS consists of files called **hosts.byname.pag** and **hosts.byname.dir**. The file with the **.pag** extension contains the key and value pairs, while the file with the **.dir** extension serves as an index for large **.pag** files.

The default NIS maps maintain values in the same format as the ASCII files from which they are derived. For example, each entry in the **/etc/passwd** file has seven fields that are separated by colons. The values in the NIS password name map have the same seven fields in the same order and format, with colons separating the fields. When programs are redirected to the password name map, they find the data in the same format as that in the **/etc/passwd** file.

Some standard NIS maps, like the password map, change frequently. To make changes to this kind of NIS map, change the ASCII files on the master server and recreate the maps.

When you modify databases on the NIS master server, you must propagate the changes from the master server to the NIS slave servers. To propagate a map means to copy it from the NIS master server to all NIS slave servers. Propagating the maps eliminates having to go to each system and update each map individually. Propagation also ensures that all copies of the database are exactly the same; therefore, any server can respond to a client's request. Having a master and several slave copies of the maps makes the NIS service highly reliable and available to its clients.

Related Information

The **makedbm** command.

Network Information Service (NIS) Overview for System Management on page 11–53, Understanding NIS Servers on page 11–53, Maintaining NIS on page 11–54.

Understanding NIS Servers

An NIS server is a host that provides resources for other computers on the network. These server computers run the **ypserv** daemon continually in order to process information for NIS maps. For any NIS map, one NIS server is designated as the *master server.* Any changes to an NIS map should be made on the master server. The changes made to the master server are then propagated to the *slave servers.*

In the NIS environment, only NIS servers have a set of NIS maps, which they make available to clients over the network.

There are two kinds of NIS servers: NIS slave servers and the NIS master server. The master server must be set up before the slave servers are set up. The master server updates the maps on the slave servers. Therefore, you should always modify the maps on the NIS master server.

A server may be a master with regard to one map and a slave with regard to another. Do not randomly assign maps to NIS servers. Use the **mkmaster** command to designate a single server as the master for all the maps you create within a single domain.

Related Information

The **mkmaster** command, **ypinit** command.

The **ypserv** daemon.

Network Information Service (NIS) Overview for System Management on page 11–47, Understanding NIS Maps on page 11–51, Maintaining NIS on page 11–54.

Maintaining NIS

The NIS environment requires adjustments from time to time. In large or complex networks, the NIS environment may change many times a day.

The following is a list of NIS maintenance tasks:

- Changing the NIS maps
- Creating an NIS map
- Updating NIS slave server maps
- Changing NIS passwords
- Adding a new NIS slave server.

A majority of the NFS and NIS maintenance commands let the user specify when the requested action takes effect. The three choices usually given are `system restart`, `now`, or `both`. The choice of `system restart` makes the necessary changes to the system so the changes occur on the next system restart. The choice of `now` makes the requested changes immediately, but makes no permanent changes to the system. The choice of `both` combines the `system restart` and `now` choices so that the changes requested take place immediately and for every subsequent system restart.

Changing the NIS Maps

Changing the NIS maps to reflect updated system information may be a common occurrence in your daily maintenance tasks. System information, such as a new user account or a changed password, can require constant updating. To modify most NIS maps, edit the text input file on the NIS master server that contains the original NIS map, rebuild the NIS map, and then propagate it to the NIS slave servers. The special handling of NIS password maps is the exception. See How to Change an NIS Map on page 11–96.

Warning: Except for user password changes made with the **yppasswd** command, NIS maps should be modified on the NIS master server only. Modifying NIS maps on NIS slave servers can break the NIS service algorithm, which can result in unreliable NIS map data.

Creating an NIS Map

User information requirements at your site may make it necessary to add new maps to your NIS domain. Standard text files can be created or filtered through the **awk**, **grep**, or **sed** commands, and passed as input to the **makedbm** utility that creates the NIS maps. It is recommended that you use mechanisms similar to those in the **/etc/yp/Makefile** file when creating the new maps. See How to Create an NIS Map from a Text File on page 11–98.

Updating NIS Slave Server Maps

To ensure that the information in NIS maps is reliable and consistent throughout an NIS domain, all updates to the maps located on the NIS master server must be propagated to the NIS slave servers. When a map on the NIS master server is updated, the **make** command automatically executes the **yppush** command after rebuilding the map.

The **yppush** command notifies all NIS slave servers that a map must be transferred. The **ypserv** daemon on each NIS slave server invokes the **ypxfr** command to get the updated map. An NIS slave server that is out of service when the **yppush** command is invoked retains the earlier version of the NIS map when it returns to the network.

To prevent such situations, use the **cron** daemon to set each NIS slave server to request updated maps from the NIS master server at regular intervals. See How to Update an NIS Slave Server Map on page 11–100.

Changing NIS Passwords

Users can change their password using the AIX **passwd** command if they are logged in on the NIS master server and the **/etc/passwd** file on the NIS master server is being used for NIS map input. If the **yppasswdd** daemon is running on the NIS master server, users may change their password from any host in the NIS domain by using the **yppasswd** command. See How to Change NIS Passwords on page 11–97.

Note: The **yppasswdd** daemon, which runs on the NIS master server only, is usually invoked when the system starts. See How to Configure the NIS Master Server on page 11–72.

Adding a New NIS Slave Server

If your network configuration grows or changes, it may be necessary to add additional NIS slave servers to support the new configuration. Adding a new NIS server to your configuration involves modifying the **ypservers** map. This map is also used by the system to determine which NIS hosts receive the replicas of the NIS maps on the NIS master server. See How to Add a New NIS Slave Server on page 11–93.

Related Information

The **chypdom** command, **make** command, **yppasswd** command, **yppush** command, **ypxfr** command.

The **yppasswdd** daemon.

Network Information Service (NIS) Overview for System Management on page11–47.

How to Change a New NIS Master Server on page 11–95.

Understanding NIS Maps on page 11–51, Understanding NIS Servers on page 11–53.

Changing Security with NIS

Security on a system running NIS is dependent on how NIS consults the administrative files on which the maps are based. In some cases, the text file is consulted first, and then the NIS maps are consulted if the NIS escape sequence is present. These files are the **/etc/passwd** and **/etc/group** files, and the **/etc/host.equiv** and **/etc/.rhosts** files.

The remaining files on which NIS maps are based are the following global files:

- **/etc/hosts**
- **/etc/networks**
- **/etc/ethers**
- **/etc/services**
- **/etc/netmasks**
- **/etc/protocols**
- **/etc/netgroup**

The information in these files is network-wide data, accessed only from NIS. However, when booting, each machine needs an entry for itself in the **/etc/hosts** file. In summary, if NIS is running, global files are checked in the NIS maps only; a global file on your local machine is not consulted.

Special NIS Password Change

When you change your password with the **passwd** command, you change the entry explicitly given in your machine's local **/etc/passwd** file. If your password is not given explicitly, but rather is pulled in from NIS with an entry, then the **passwd** command prints the following error message:

```
Not in passwd file.
```

To change your password in the NIS **password** file, use the **yppasswd** command. To enable this service, start up the **yppasswdd** server daemon on the machine serving as the master server for the NIS password file.

The /etc/publickey File

Before users can take advantage of the secure option to mount a directory, the NIS **publickey** map must exist. The **publickey** map is used for secure networking. Each entry in the map consists of a network user name (which may refer to either a user or a host name), followed by the user's public key (in hexadecimal notation), a colon, and then the user's secret key encrypted with its **keylogin** password (also in hexadecimal notation). This file can be altered with either the **chkey** command or the **newkey** command. The **/etc/publickey** file should only contain data on the NIS master machine, where it is converted into the **publickey.byname** NIS map.

Enter the following command to create the NIS database from the **/etc/publickey** file:

```
cd /etc/yp
make publickey
```

Do not use a text editor to alter the **/etc/publickey** file because the file contains encryption keys. To alter the **/etc/publickey** file, use the **newkey** command. This command has two options. For a regular user on a host machine, enter:

```
newkey –u username
```

For a root user on a host machine, enter:

```
newkey -h hostname
```

The **publickey** database contains every user with a public key. They are identified by a long string. Users can create keys for themselves with the **chkey** command.

You should also be aware that there is a new NIS database called **netid**; however, you do not need to administer it. The **netid** database is created from the **passwd**, **host**, and **group** files.

Netgroups: Network-Wide Groups of Machines and Users

NIS uses the **/etc/netgroup** file on the master NIS server for permission checking during remote mount, login, remote login, and remote shell. It uses the **/etc/netgroup** file to generate three NIS maps in the **/etc/yp/chypdom** directory: **netgroup**, **netgrp.usr**, and **netgrp.hst**. The NIS map netgroup contains the basic information in the **/etc/netgroup** file. The two other NIS maps contain a more specific form of the information to speed up the process of looking up netgroups given the host or user.

The programs that consult these NIS maps are the **login** command, the **mountd** daemon, the **rlogin** command, and the **rsh** command. The **login** command consults the maps for user classifications if it encounters netgroup names in the **/etc/passwd** file. The **mountd** daemon consults the maps for machine classifications if it encounters **netgroup** names in the **/etc/exports** file. The **rlogin** command and the **rsh** command consult the **netgroup** map for both machine and user classifications if they encounter **netgroup** names in the **/etc/hosts.equiv** or **/.rhosts** files.

Related Information

The **chkey** command, **newkey** command, **rlogin** command, **rsh** command, **passwd** command, **keylogin** command, **login** command.

The **ypupdated** daemon, **mountd** daemon, **yppasswdd** daemon.

Network Information Service (NIS) Overview for System Management on page 11–47.

Understanding NIS Maps on page 11–51, Maintaining NIS on page 11–54.

Understanding NIS Services

Troubleshooting NIS–Related Problems

The approach to troubleshooting an NIS problem depends on whether the problem is at the NIS client or the NIS server.

Identifying NIS Client Problems

The most common NIS client problems occur at the following times:

- When commands hang
- When NIS service is unavailable
- When the ypbind daemon fails
- When the ypwhich command is inconsistent.

When Commands Hang

The most common problem occurring at an NIS client node is for a command to hang. Sometimes a command appears to hang, even though the system seems fine and other commands run. In such a case, a message like the following may be generated at the console:

```
NIS: server not responding for domain <wigwam>. Still trying.
```

This error message indicates that the **ypbind** daemon on the local machine is unable to communicate with the **ypserv** daemon in the `wigwam` domain. This results when systems that run the **ypserv** daemon have failed. It may also occur if the network or the NIS server machine is so overloaded that the **ypserv** daemon cannot get a response back to your **ypbind** daemon within the time-out period.

Under these circumstances, all the other NIS clients on your network show the same or similar problems. The condition is usually temporary. The messages go away when the NIS server machine reboots and the **ypserv** daemon restarts, or else when the load on the NIS server and the network decreases.

If the **ypbind** daemon is communicating with the **ypserv** daemon and the NIS server is not overloaded, one of the following problems may exist:

- The `domainname` on the NIS client machine is not set or is set incorrectly. Clients must use a domain name that the NIS servers know. In this case, set the domain correctly.
- Your local network may not have an NIS server machine. You can automatically bind only to a **ypserv** process on your local network. Binding on servers residing on other networks through a gateway requires using the **ypset** command. Two or more NIS servers (one master and several slave servers) can improve availability and response characteristics for NIS services.
- The NIS server may not be up and running. Check other machines on your local network. If several clients have problems simultaneously, the server may be the cause.

  Find a client system that is working normally and try the **ypwhich** command. If the **ypwhich** command never returns an answer, stop it. Then type the following at the NIS server machine:

  ```
  ps -ef | grep yp
  ```

Look for the **ypserv** and **ypbind** processes. If the server's **ypbind** daemon is not running, start it by typing the following:

```
startsrc -s ypbind
```

If a **ypserv** process is running, issue the **ypwhich** command on the NIS server machine. If this command returns no answer, the **ypserv** daemon is probably hung and should be restarted.

To kill the existing **ypserv** process, log on as root and type the following:

```
stopsrc -s ypserv
```

If this shows no **ypserv** process running, start one by typing the following:

```
startsrc -s ypserv
```

When NIS Service Is Unavailable

When other machines on the network appear to have no problems, but NIS service becomes unavailable on your system, a variety of symptoms can show up:

- Some commands may operate correctly while others terminate and print an error message about the unavailability of NIS.
- Some commands limp along in a backup-strategy mode particular to the program involved.
- Some commands or daemons crash with obscure messages or no message at all.

For example, messages such as the following may show up:

```
ypcat myfile
ypcat: can't bind to NIS server for domain <wigwam>
Reason: can't communicate with ypbind.
```

OR

```
/usr/etc/yp/yppoll myfile
RPC: timed out
```

When symptoms like these occur, issue the **ls -l** command on a directory containing files owned by many users, including users not in the local machine's **/etc/passwd** file. Use the following format:

```
ls -l
```

If the **ls -l** command reports file owners that are not in the local machine's **/etc/passwd** file as numbers, rather than names, it means that NIS service is not working.

These symptoms usually indicate that your **ypbind** daemon is not running. You can use the **ps -ef** command to check for one.

If you do not find a **ypbind** daemon, type the following to start it:

```
startsrc -s ypbind
```

When the ypbind Daemon Becomes Inoperable

If the **ypbind** daemon repeatedly crashes immediately after it is started, you should look for a problem in some other part of the system.

- Check for the presence of the **portmap** daemon by typing:

  ```
  ps –ef | grep portmap
  ```

  If it is not running, reboot the system.

- If the **portmap** daemon is running but will not stay up or responds strangely, check the network software.

  Try to communicate with the **portmap** daemon on your machine from a different machine that is operating normally. From such a machine, type:

  ```
  rpcinfo –p client
  ```

- If the **portmap** daemon is up and running, the output appears in a format similar to the following:

```
program   vers proto   port

100007      2    tcp   1024  ypbind
100007      2    udp   1028  ypbind
100007      1    tcp   1024  ypbind
100007      1    udp   1028  ypbind
100021      1    tcp   1026  nlockmgr
100024      1    udp   1052  status
100020      1    udp   1058  llockmgr
100020      1    tcp   1028  llockmgr
100021      2    tcp   1029  nlockmgr
100012      1    udp   1083  sprayd
100011      1    udp   1085  rquotad
100005      1    udp   1087  mountd
100008      1    udp   1089  walld
100002      1    udp   1091  rusersd
100002      2    udp   1091  rusersd
100001      1    udp   1094  rstatd
100001      2    udp   1094  rstatd
100001      3    udp   1094  rstatd
```

If the daemons are not listed, the **ypbind** daemon is unable to register its services. Reboot the machine.

If the daemons are there, but they change each time you try to restart the **/usr/etc/ypbind** daemon, reboot the system (even though the **portmap** daemon is up).

When the ypwhich Command Is Inconsistent

When you use the **ypwhich** command several times at the same client node, the response varies, because the status of the NIS server changes. This is normal.

The binding of NIS client to NIS server changes over time on a busy network, when the NIS servers are busy. Whenever possible, the system stabilizes so that all clients get acceptable response time from the NIS servers. The source of an NIS service is not important, because an NIS server machine often gets its own NIS services from another NIS server on the network.

Identifying NIS Server Problems

The most common NIS server problems occur at the following times:

- When different versions of an NIS map exist
- When the **ypserv** daemon crashes.

When Different Versions of an NIS Map Exist

Since NIS works by propagating maps among servers, you can sometimes find different versions of a map at the network servers. This is normal if temporary and abnormal otherwise.

Normal update is prevented when an NIS server or a router between NIS servers is down during a map transfer attempt. When all the NIS servers and all the routers between them are up and running, the **ypxfr** command should succeed.

If a particular slave server has problems updating a map, you can log in to that server and run the **ypxfr** command interactively. If this command fails, an error message returns to tell you why, so that you can fix the problem. If the command succeeds, but you think it has been failing sometimes, create a log file to enable logging of messages by typing the following:

```
cd /etc/yp
touch ypxfr.log
```

This saves all output from the **ypxfr** command. The output looks much like what the **ypxfr** command creates when it is run interactively, but each line in the log file is time stamped. The time stamp tells when the **ypxfr** command began its work. It is normal to see unusual orderings in the time stamps. If copies of the **ypxfr** command ran simultaneously but their work took differing amounts of time, the summary status line may be written to the log files in an order that differs from the order in which they were invoked.

Any pattern of intermittent failure shows up in the log. After you fix the problem, turn off logging by removing the log file. If you forget to remove the log file, it grows without limit.

While you are logged in to the NIS slave server, inspect the system **crontab** entries, and the **ypxfr** shell scripts it invokes.

Make sure that the NIS slave server is in the **ypservers** map. If not, the **yppush** command will not notify the slave server when a new copy of a map exists.

When the ypserv Daemon Becomes Inoperable

When the **ypserv** process repeatedly crashes immediately after starting and does not stay up with repeated activations, the debugging process is similar to that described for **ypbind** crashes. First, you should check for the **portmap** daemon:

```
ps –ef | grep portmap
```

If you do not find the **portmap** daemon, reboot the server.

If there is a **portmap** daemon, type:

```
rpcinfo –p speed
```

On your particular machine, the port numbers will be different. The four entries that represent the **ypserv** daemon are:

```
100004      2    udp    1027  ypserv
100004      2    tcp    1024  ypserv
100004      1    udp    1027  ypserv
100004      1    tcp    1024  ypserv
```

If these entries do not exist, the **ypserv** daemon is unable to register its services. Reboot the machine. If the **ypserv** entries are present, but they change each time you try to restart the **/usr/etc/ypserv** daemon, reboot the machine again.

Related Information

Network Information Service (NIS) Overview for System Management on page 11–47.

Troubleshooting NFS–Related Problems on page 11–37.

How to Configure NFS

Prerequisite Tasks or Conditions

1. Install and configure TCP/IP.
2. Install NFS as described in How to Install AIX for RISC System/600 Licensed Program Optional Program Products in *General Concepts and Procedures*.
3. Decide which machines on the network will be exporting directories (servers) and which machines will be mounting those directories (clients).

Procedure

This procedure summarizes the configuration process. See Configuring NFS on Your System on page 11–25 for more information on this procedure.

1. Create the **/etc/exports** file for each system that is designated as an NFS server. See How to Create and Edit the /etc/exports File for NFS on page 11–77 and How to Configure an NFS Server on page 11–66 for details on performing these tasks.
2. For each system that is designated as an NFS client, do the following:
   a. Edit the **/etc/vfs** file to make NFS the default remote file system.
   b. Establish the local mount points using the **mkdir** command. See How to Configure an NFS Client on page 11–65 for details on performing these tasks.
3. Configure the system to start NFS. This can be done one of three ways:
   a. Use the **smit mknfs** command to start NFS. This takes advantage of the SMIT interface to start NFS on your system. The menu that is presented after invoking this command asks `START NFS now, on system restart or both?`. If you choose `now`, the **/etc/rc.nfs** script is executed to start the NFS daemons. If you choose `system restart`, an entry is placed in the **inittab** file so that the **/etc/rc.nfs** script is executed on the next system restart. If you choose `both`, then the **/etc/rc.nfs** script is executed now and an entry is placed in the **inittab** file so that the **/etc/rc.nfs** script is executed on the next system restart.
   b. Use the **mknfs** command to start NFS. The **mknfs** command is the command that is executed from the SMIT menu described above. Given the correct parameter, the **mknfs** command accomplishes the same effects as described in the procedure above. See the **mknfs** command documentation for more information.
   c. Edit the appropriate files to start NFS:
      1. Edit the **/etc/rc.tcpip** file and make sure that the entry to start the **portmap** daemon is uncommented.
      2. Edit the **/etc/rc.nfs** file and make sure that the entries for the **nfsd** and **biod** daemons are uncommented.
      3. Add an entry to the **inittab** file so that the **/etc/rc.nfs** script is executed on system restart. Use the **mkitab** command to add an entry to the **inittab file**.

   See How to Configure NFS Servers and Clients on page 11–67 for details on performing these tasks.
4. Configure PC-NFS for communication with personal computers, if necessary.
   a. Modify the **inetd** daemon configuration database to enable the starting of the **pcnfsd** daemon.
   b. Use the **rpcinfo** command to verify the **pcnfsd** daemon is active.

   See How to Configure the inetd Daemon on page 14–102 for details on updating the **inetd** configuration database.

Related Information

The **mknfs** command, **smit** command.

The **biod** daemon, **inetd** daemon, **mountd** daemon, **nfsd** daemon, **pcnfsd** daemon, **portmap** daemon.

The **/etc/rc.tcpip** file, **inittab** file.

How to Configure the inetd Daemon on page 14–102, How to Configure an NFS Server on page 11–66, How to Configure an NFS Client on page 11–65.

Installing NFS on page 11–24, Configuring NFS on Your System on page 11–25, Configuring an NFS Server on page 11–25, Configuring an NFS Client on page 11–26, Configuring PC-NFS on page 11–29.

Understanding Protocols for TCP/IP on page 14–54.

How to Configure an NFS Client

Prerequisite Tasks or Conditions

1. Install and configure TCP/IP.
2. Install NFS as described in How to Install AIX for RISC System/600 Licensed Program Optional Program Products in *General Concepts and Procedures*.
3. Decide which machines on the network will be exporting directories (servers) and which machines will be mounting those directories (clients).

Procedure

Configure each system that is designated as an NFS client.

1. Make NFS the default remote file system. If this is not done, it is necessary to specify the **–v** flag when doing mounts. Using a text editor, open the **/etc/vfs** file and search for the following entry:

   ```
   #%defaultvfs jfs nfs

   #nfs 2 /etc/helpers/nfsmnthelp none remote
   ```

   If pound signs (#) appear at the beginning of the lines as shown, delete the signs.

2. Start NFS by using the **smit mknfs** command. Other methods are available. See How to Configure NFS Clients and Servers on page 11–67 for details on this procedure.
3. Use the **mkdir** command to establish the local mount points. The mount points for all NFS mounts must exist on your system.

Note: Note that there are exceptions to this rule. If the **automount** daemon is used it may not be necessary to create mount points. See the **automount** documentation for details of its use.

4. Mount the directories. This can be done by three different methods.
   a. Use the **smit mknfsmnt** command to create an NFS mount. Enter the required data for this menu. This method creates an entry in the **/etc/filesystems** file for the desired mount and also attempts the mount.
   b. Use the **mknfsmnt** command directly. This command is the one that is executed from the SMIT menu described above. Given the correct parameters, this command accomplishes the same tasks as it would if it were executed using SMIT.
   c. Edit the **/etc/filesystems** file and add a stanza for the file system to be mounted.

   See How to Mount a Directory Using NFS on page 11–79 for details.

Related Information

The **mkdir** command, **mknfsmnt** command.

The **biod** daemon.

The **/etc/filesystems** file, **/etc/vfs** file.

Installing NFS on page 11–24, Configuring an NFS Server on page 11–25, Configuring an NFS Client on page 11–26, Configuring PC-NFS on page 11–29.

Understanding Protocols for TCP/IP on page 14–54.

How to Configure an NFS Server

Prerequisite Tasks or Conditions

1. Install and configure TCP/IP.
2. Install NFS as described in How to Install AIX for RISC System/600 Licensed Program Optional Program Products in *General Concepts and Procedures*.
3. Decide which machines on the network will be exporting directories (servers) and which machines will be mounting those directories (clients).

Procedure

Configure each system that is designated as an NFS server:

1. Create the **/etc/exports** file. This can be done by three different methods.
   a. Using a text editor, open a new file named **/etc/exports**. List by full path name each directory that can be exported in this file, aligning each directory name at the left margin.
   If you have not determined which directories are to be exported, the file can still be created. The presence of the file is enough to start the **nfsd** daemons. The **/etc/rc.nfs** file checks for the existence of this file, but doesn't check the contents. If the file exists, the **nfsd** daemons and the **mount** daemon are started.

   See How to Create and Edit the /etc/exports File for NFS on page 11–77 for details.

   b. Using the **smit mknfsexp** command, fill in the menu with the appropriate data. This exports the directory specified. It is important to note that if this method is used, NFS should already be running. If the **/etc/exports** file does not exist, then it will be created. Also note that if the **/etc/exports** directory does not exist before the creation of an export, then the **nfsd** and **mount** daemons are probably not running. They can be started with the following commands:
   ```
   startsrc -s nfsd
   ```
   AND
   ```
   startsrc -s rpc.mountd
   ```
   This should only be done if NFS is not currently running.
   c. Use the **mknfsexp** command directly from the command line. This command is executed by the SMIT menu described above, and has the same effects.
2. Start NFS by one of the following methods.
   a. Use the **smit mknfs** command and choose the appropriate time to start NFS.
   b. Use the **mknfs** command directly to start NFS.
   c. Edit the **/etc/rc.tcpip** file to start the **portmap** and **inetd** daemons, and then add an entry to the **inittab** file to start NFS on system restart.
   See How to Configure NFS Clients and Servers on page 11–67 for details.

Related Information

The **inetd** daemon, **mountd** daemon, **nfsd** daemon.

The **/etc/exports** file, **/etc/inetd.conf** file.

Installing NFS on page 11–24, Configuring an NFS Server on page 11–25, Configuring an NFS Client on page 11–26, Configuring PC-NFS on page 11–29.

Understanding Protocols for TCP/IP on page 14–54.

How to Configure NFS Clients and Servers

Prerequisite Tasks or Conditions

1. Install and configure TCP/IP.
2. Install NFS as described in How to Install AIX for RISC System/600 Licensed Program Optional Program Products in *General Concepts and Procedures*.
3. Decide which machines on the network will be exporting directories (servers) and which machines will be mounting those directories (clients).
4. Perform the configuration tasks that apply to either the NFS server or clients as described in How To Configure an NFS Server on page 11–66 and How to Configure an NFS Client on page 11–65.

Procedure

Configure each system to run NFS.

1. Make sure that the appropriate procedures have been completed for the NFS server or the NFS client.
2. Configure the system to start NFS. This can be done one of three ways:
   a. Use the **smit mknfs** command to start NFS. This command is the SMIT interface for starting NFS on your system. The menu that is presented to the user after invoking this command asks `START NFS now, on system restart or both?`. If the user chooses `now`, the **/etc/rc.nfs** script is executed to start the NFS daemons. If the user chooses `system restart`, an entry is placed in the **inittab** file so that the **/etc/rc.nfs** script is executed on the next system restart. If the user chooses `both`, then the **/etc/rc.nfs** script is executed now and an entry is placed in the **inittab** file so that the **/etc/rc.nfs** script is executed on the next system restart.
   b. Use the **mknfs** command to start NFS. The **mknfs** command is the same command as the one executed from the SMIT menu. Given the correct parameter, the **mknfs** command accomplishes the same effects as the above procedure. See the **mknfs** command documentation for more information.
   c. Edit the appropriate files to start NFS.
      - Edit the **/etc/rc.tcpip** file and make sure that the **portmap** and **inetd** daemons are started. Edit the **/etc/rc.tcpip** file and search for the following entries:

```
# Start up Portmapper
#if [ -f /usr/etc/portmap ]; then
#         startsrc -s portmap
#fi
```

AND

```
# Start up socket based daemons
#if [ -f /etc/inetd ]; then
#        startsrc -s inetd
#fi
```

If pound signs (#) appear at the beginning of the lines as shown, delete the signs.

Note: In the **/etc/rc.tcpip** file, the **portmap** daemon entry must appear before the **inetd** daemon entry.

- Add an entry to the **inittab** file so that the **/etc/rc.nfs** file is executed on system restart. Use the **mkitab** command to add the entry. For example:

```
mkitab "nfsyp:2:wait:/etc/rc.nfs > /dev/console 2>&1"
```

3. Update the **inetd** configuration database so that the desired RPC daemons are started. Look for the following lines in the **/etc/inetd.conf** file and follow the procedures described in How to Configure the inetd Daemon on page 14–102.

```
#rexd sunrpc_tcp tcp wait root /usr/etc/rpc.rexd rexd 100017 1
#rstatd sunrpc_udp udp wait root /usr/etc/rpc.rstatd rstatd
100001 1–3
#rusersd sunrpc_udp udp wait root /usr/etc/rpc.rusersd rusersd
100002 1–2
#rwalld sunrpc_udp udp wait root /usr/etc/rpc.rwalld rwalld
100008 1
#sprayd sunrpc_udp udp wait root /usr/etc/rpc.sprayd sprayd
100012 1
#pcnfsd sunrpc_udp udp wait root /etc/rpc.pcnfsd pcnfsd 150001
1
```

Related Information

The **mknfs** command, **mkitab** command.

The **inetd** daemon, **pcnfsd** daemon, **portmap** daemon.

The **/etc/inetd.conf** file, **/etc/rc.tcpip** file.

How to Configure the inetd Daemon on page 14–102.

Installing NFS on page 11–24, Configuring an NFS Server on page 11–25, Configuring an NFS Client on page 11–26.

Understanding Protocols for TCP/IP on page 14–54.

How to Configure PC–NFS

Prerequisite Tasks or Conditions

1. Install and configure TCP/IP.
2. Install NFS as described in How to Install AIX for RISC System/600 Licensed Program Optional Program Products in *General Concepts and Procedures*.
3. Decide which machines on the network will be exporting directories (servers) and which machines will be mounting those directories (clients).
4. Configure the NFS servers as described in How to Configure an NFS Server on page 11–66.

Procedure

If your network includes IBM Personal Computers that are running versions of the Disk Operating System (DOS) and have the PC-NFS program installed, you can configure the **pcnfsd** daemon on NFS servers to allow users on the PC-NFS clients to access authentication and print-spooling services.

1. Select a suitable location for the spool directories, such as the **/usr/tmp** directory. The spool directory must have at least 100K bytes of free space.
2. Export the directory containing the spool directories. Do not put access restrictions on the exported directory that could cause access problems in your network. See How to Export a Directory Using NFS on page 11–79 for details of this procedure.
3. Start the **pcnfsd** daemon to use the default spool directory, which is the **/usr/tmp** directory.

   - Modify the **inetd** configuration database so that the **pcnfsd** daemon is started. The following entry in the **/etc/inetd.conf** file corresponds to the **pcnfsd** daemon:

     ```
     pcnfsd sunrpc_udp udp wait root /etc/rpc.pcnfsd pcnfsd 150001 1
     ```

     Note: The default directory that the **pcnfsd** daemon uses cannot be changed from the **inetd.conf** file.

     See How to Configure the inetd Daemon on page 14–102 for details on updating the **inetd** configuration database.

     OR

   - Start the **pcnfsd** daemon to use a directory other than the default spool directory. Identify the directory to use as the spooling directory by adding the following entry to the **/etc/rc.nfs** file:

     ```
     if [ –f /etc/rpc.pcnfsd ] ; then
     /etc/rpc.pcnfsd –s spooldir ; echo ' rpc.pcnfsd\c'
     fi
     ```

     The *spooldir* parameter specifies the name of the directory to be assigned as the **pcnfsd** daemon print spooling directory.

     Start the **pcnfsd** daemon print spooler by entering the following at the command line:

     ```
     /etc/rpc.pcnfsd –s spooldir
     ```

Deactivate the **pcnfsd** print-spooling directory default entry in the **/etc/inetd.conf** file. See How to Configure the inetd Daemon on page 14–102 for details of updating the **inetd** configuration database so that the **pcnfsd** daemon is not started by the **inetd** daemon.

4. Verify that the **pcnfsd** daemon is accessible. If you start the **pcnfsd** daemon from the command line, verify that the daemon is accessible through the network by entering the following at the command line:

   ```
   rpcinfo –u localhost 150001 1
   ```

   The `localhost` parameter specifies the host name of the system on which you are configuring **pcnfsd**. After you have entered the command, you should receive the message that the program is ready and waiting.

Related Information

The **inetd** daemon, **pcnfsd** daemon.

The **/etc/inetd.conf** file.

How to Configure the inetd Daemon on page 14–102.

Installing NFS on page 11–24, Configuring an NFS Server on page 11–25, Configuring an NFS Client on page 11–26, Configuring PC-NFS on page 11–29.

Understanding Protocols for TCP/IP on page 14–54.

How to Configure NIS

Prerequisite Tasks or Conditions

1. NFS must be installed on your system.
2. TCP/IP must be running.
3. The **portmap** daemon must be running.

Procedure

1. Modify the *PATH* variable in the **.profile** file to list the **/usr/etc/yp** directory where commands for NIS can be found.
2. Set the NIS domain name on each host using the **chypdom** command, so that the NIS maps can be created and distributed in the proper domain.
3. Configure the NIS master server.
4. Configure NIS slave servers.
5. Configure NIS clients.

Related Information

The **chypdom** command, **domainname** command.

The **ypbind** daemon, **ypserv** daemon.

Network Information Service (NIS) Overview for System Management on page 11–47.

How to Configure NIS Clients on page 11–74, How to Configure the NIS Master Server on page 11–72, How to Configure NIS Slave Servers on page 11–73.

How to Configure the NIS Master Server

Prerequisite Tasks or Conditions

1. The NFS software package must be installed on your system.
2. TCP/IP must be running on your system.
3. The **portmap** daemon must be running on your system.
4. The domain name must be set.

Procedure

Perform the following tasks on the master server host:

1. Edit the NIS map input files.
2. Use the **smit mkmaster** command to create the NIS maps and start the NIS daemons that consult NIS maps.

   If you want to set up your system for secure RPC networking, enable the **ypupdated** and **yppasswdd** daemons while running the **smit mkmaster** command.
3. Configure secure NFS on your system if you are setting up your system for secure RPC networking. Start the **keyserv** daemon from the command line or use the following SMIT command:

   ```
   smit mkkeyserv
   ```

   To start the **keyserv** daemon from the command line, enter:

   ```
   mkkeyserv
   ```

An alternative method of configuring the master server is to execute the **ypinit** command to build the NIS maps:

1. Issue the **ypinit** command.
2. Start the **ypserv** and **ypbind** daemons.
3. Edit the **/etc/rc.nfs** file to uncomment the lines that use the **startsrc** commands to start these daemons.
4. Uncomment the entry in the **/etc/rc.nfs** file to start **keyserv** daemon.

Related Information

The **startsrc** command, **ypinit** command.

The **keyserv** daemon, **ypbind** daemon, **yppasswdd** daemon, **ypserv** daemon.

Network Information Service (NIS) Overview for System Management on page 11–47.

How to Configure NIS on page 11–71, How to Configure NIS Clients on page 11–74, How to Configure NIS Slave Servers on page 11–73, How to Edit the NIS Map Input Files on page 11–105, How to Start and Stop NIS Daemons on page 11–104.

Configuring Secure NFS on page 11–16.

How to Configure NIS Slave Servers

Prerequisite Tasks or Conditions

1. NFS must be installed on your system.
2. The domain name must be set.
3. TCP/IP must be running.
4. The **portmap** daemon must be running.

Procedure

Perform the following tasks on the slave server hosts:

1. Either use the **smit mkslave** command fastpath or type the **mkslave** command at the command line to set up the NIS slave server and start the **ypserv** and **ypbind** daemons.
2. Edit the NIS map input files.

An alternative method of configuring the slave server is to execute the **ypinit** command with the **-s** flag.

1. Issue the **ypinit** command.
2. Start the **ypserv** and **ypbind** daemons by editing the **/etc/rc.nfs** file.
3. Uncomment the lines that use the **startsrc** commands to start these daemons.

Related Information

The **mkslave** command, **smit** command, **startsrc** command, **ypinit** command.

The **portmap** daemon, **ypbind** daemon, **ypserv** daemon.

Network Information Service (NIS) Overview for System Management on page 11–47.

How to Configure NIS on page 11–71, How to Configure NIS Clients on page 11–74, How to Configure the NIS Master Server on page 11–72, How to Start and Stop NIS Daemons on page 11–104, How to Edit the NIS Map Input Files on page 11–105, How to Add a New NIS Slave Server on page 11–93 .

How to Configure NIS Clients

Prerequisite Tasks or Conditions

1. NFS must be installed on your system.
2. The NIS domain name must be set.
3. TCP/IP must be running.
4. The **portmap** daemon should be running on your system.
5. An NIS master server should be configured.

Procedure

Perform the following tasks on the client hosts:

1. Edit the NIS map input files to use NIS servers.
2. Start the NIS daemons that consult the NIS maps upon client request, by running the **smit mkclient** command or entering the **mkclient** command on the command line.

Another method for starting the **ypbind** daemon for client requests is to edit the **/etc/rc.nfs** file and uncomment the **startsrc** command that starts the **ypbind** daemon.

Related Information

The **mkclient** command, **smit** command, **ypbind** command.

Network Information Service (NIS) Overview for System Management on page 11–47.

How to Configure NIS on page 11–71, How to Configure the NIS Master Server on page 11–72, How to Configure NIS Slave Servers on page 11–73, How to Start and Stop NIS Daemons on page 11–104.

How to Start and Stop the NFS Daemons Using SRC

Prerequisite Tasks or Conditions

1. Install and configure TCP/IP.
2. Install NFS as described in How to Install AIX for RISC System/600 Licensed Program Optional Program Products in *General Concepts and Procedures*.
3. Decide which machines on the network will be exporting directories (servers) and which machines will be mounting those directories (clients).
4. Configure the NFS servers as described in How to Configure an NFS Server on page 11–66.
5. Configure the NFS clients as described in How to Configure an NFS Client on page 11–65.

Procedure

The NFS daemons can be started and stopped individually or all at once using the System Resource Controller (SRC).

- To start and stop NFS daemons individually, use the following commands at the command line:

  To start the **nfsd** daemons, enter the **startsrc –s nfsd** command.
  To stop the **nfsd** daemons, enter the **stopsrc –s nfsd** command.

  To start the **biod** daemons, enter the **startsrc –s biod** command.
  To stop the **biod** daemons, enter the **stopsrc –s biod** command.

  To start the **rpc.lockd** daemon, enter the **startsrc –s rpc.lockd** command.
  To stop the **rpc.lockd** daemon, enter the **stopsrc –s rpc.lockd** command.

  To start the **rpc.statd** daemon, enter the **startsrc –s rpc.statd** command.
  To stop the **rpc.statd** daemon, enter the **stopsrc –s rpc.statd** command.

  To start the **rpc.mountd** daemon, enter the **startsrc –s rpc.mountd** command.
  To stop the **rpc.mountd** daemon, enter the **stopsrc –s rpc.mountd** command.

  To start the **portmap** daemon, enter the **startsrc –s portmap** command.
  To stop the **portmap** daemon, enter the **stopsrc –s portmap** command.

- To start and stop all of the NFS daemons at once, use the following commands at the command line:

  To start all of the NFS daemons, enter the **startsrc –g nfs** command.
  To stop all of the NFS daemons, enter the **stopsrc –g nfs** command.

- To change the parameters that are passed to any of the daemons that are started by SRC commands, enter the following at the command line:

  ```
  chssys –s nfsd –a '10'
  ```

  This command changes the parameter passed to the **nfsd** daemon to 10.

- To get the current status of all of the daemons that are controlled by SRC, enter the following at the command line:

  ```
  lssrc –a
  ```

Related Information

The **biod** daemon, **lockd** daemon, **mountd** daemon, **nfsd** daemon, **portmap** daemon, **srcmstr** daemon, **statd** daemon.

Configuring NFS on Your System on page 11–25.

System Resource Controller Overview in *General Concepts and Procedures*.

How to Create and Edit the /etc/exports File for NFS

Prerequisite Tasks or Conditions

1. Install and configure TCP/IP.
2. Install NFS as described in How to Install AIX for RISC System/600 Licensed Program Optional Program Products in *General Concepts and Procedures*.
3. Decide which machines on the network will be exporting directories (servers), and which machines will be mounting those directories (clients).

Procedure

This procedure requires that the user have root privileges:

1. Edit the **/etc/exports** file with a text editor.
2. Create an entry for each directory to be exported. To specify the directory to be exported, use the full path name of the directory. See the **/etc/exports** file documentation for a description of the full syntax for entries in the **/etc/exports** file.
3. Save and close the **/etc/exports** file.
4. If NFS is currently running, execute the **exportfs** command as follows:

   ```
   /usr/etc/exportfs -a
   ```

   The **-a** option tells the **exportfs** command to send all information in the **/etc/exports** file to the kernel.

Related Information

The **chnfsexp** command, **exportfs** command, **mknfsexp** command, **rmnfsexp** command.

The **/etc/exports** file.

Administering Secure NFS on page 11–11.

How to Use the NFS Secure Option When Exporting a Directory

Prerequisite Tasks or Conditions

1. Make sure the directory to be exported exists on the system.
2. Make sure that the **publickey** map exists and that the **keyserv** daemon is running. See Configuring Secure NFS on page 11–25 for details on these prerequisites.

Procedure

The procedure for exporting directories and using the secure option is the same as for regular exporting, except for a difference in the choice of parameters. See How to Export a Directory Using NFS on page 11–79 for details on exporting, and add the steps below for exporting and using the secure option.

Exporting can be done in one of three ways as explained in How to Export a Directory Using NFS on page 11–79. There are three ways to specify the secure option, each corresponding to one of the three methods of exporting:

1. If using the SMIT interface (the **smit mknfsexp** command) for exporting a directory, change the default `no` to `yes` for the selection `Use Secure Option?`.
2. If using the **mknfsexp** command, use the `–s` option to specify that the secure option is to be used when exporting the specified directory.
3. If editing the **/etc/exports** file to add an entry, use the `–secure` option for the new entry. This option can also be used directly from the command line, as in this example:

```
exportfs –i –o secure /directory/to/export
```

Related Information

The **chnfsexp** command, **exportfs** command, **mknfsexp** command, **rmnfsexp** command, **smit** command.

The **keyserv** daemon, **nfsd** daemon.

The **/etc/exports** file, **/etc/publickey** file.

How to Create and Edit an NFS /etc/exports File on page 11–77, How to Export a Directory Using NFS on page 11–79.

Configuring Secure NFS on page 11–16, Administering Secure NFS on page 11–29, Maintaining an NFS Server on page 11–31.

How to Export a Directory Using NFS

Prerequisite Tasks or Conditions

1. Install and configure TCP/IP.
2. Install NFS as described in How to Install AIX for RISC System/600 Licensed Program Optional Program Products in *General Concepts and Procedures*.
3. Decide which machines on the network will be exporting directories.
4. Perform the configuration procedures described in How to Configure NFS Servers and Clients on page 11–67.
5. Make sure that NFS is currently running on your system.

Procedure

This procedure describes how to export a directory using NFS so that NFS clients may mount and use the exported directory. A directory can be exported in one of the following ways:

1. Use the **mknfsexp** command. This method uses the SMIT interface for exporting directories with NFS.
2. Invoke the **mknfsexp** command directly from the command line. The SMIT interface in the first method uses the **mknfsexp** command to export the requested directory. If the user supplies the correct command line parameters, the **mknfsexp** command may be used from the command line as well.
3. Edit the **/etc/exports** file and add an entry for the directory. After the **/etc/exports** file has been updated, then execute the **exportfs** command to export the directory. See the **exportfs** command documentation for more information on the parameters for exporting directories using NFS.
4. If the **mountd** daemon and the **nfsd** daemons are not currently running on the system, they must be started in order for the exported directories to be available to the NFS clients. To start the **nfsd** daemons and the **mountd** daemon, use the following commands:

   ```
   startsrc -s nfsd
   ```

   AND

   ```
   startsrc -s rpc.mountd
   ```

Related Information

The **chnfsexp** command, **exportfs** command, **mknfsexp** command, **rmnfsexp** command, **smit** command.

The **mountd** daemon, **nfsd** daemon.

The **/etc/exports** file.

How to Create and Edit the NFS /etc/exports File on page 11–77, How to Use the NFS Secure Option When Exporting a Directory on page 11–78.

Administering Secure NFS on page 11–11, Maintaining an NFS Server on page 11–31.

How to Mount a File System Using NFS

Prerequisite Tasks or Conditions

1. Make sure that the system that is to be the server is currently exporting the directory that will be mounted on the local system. To do this, use the **showmount** command:

   ```
   showmount -e ServerName
   ```

   This shows the directories that are currently exported from the system specified by the *ServerName* parameter.

2. Make sure that a mount point exists on the client system for the file system that is to be mounted. The mount point may be created with the **mkdir** command.

Procedure

NFS clients mount directories from NFS servers as if they were local file systems. Issuing the **mount** command from the NFS client logically binds the client to the server.

Mount another system's exported directory as a local file system in one of the following three ways:

1. Use the **smit mknfsmnt** command. This method uses the SMIT interface for mounting file systems from NFS servers. Enter the required data and options and then execute the mount using SMIT.
2. Invoke the **mknfsmnt** command directly from the command line. The SMIT interface in the first method uses the **mknfsmnt** command to mount the requested file system. If invoked with the correct parameters, the **mknfsmnt** command accomplishes the same task as the SMIT interface.
3. Invoke the **mount** command from the command line with the correct options in order to mount a file system from a server. For example:

   ```
   mount ServerName:/remote/directory /local/directory
   ```

   In this example, the **mount** command mounts the `/remote/directory` directory from the NFS server specified by the *ServerName* parameter over the `/local/directory` directory on the NFS client.

   Included here are further examples of how the **mount** command might be used:

   - To mount all NFS file systems defined in the **/etc/filesystems** file, enter the following command:
     ```
     mount -v nfs all
     ```
   - To mount a specific NFS file system defined in the **/etc/filesystems** file, enter the following command:
     ```
     mount dirname
     ```
     Replace the *dirname* parameter with the name of the directory to be mounted.
   - To mount several NFS file systems defined in the **/etc/filesystems** file, enter the following command:
     ```
     mount -t type_name
     ```

Replace the *type_name* parameter with the name of the group of files to be mounted. Note that the *type_name* parameter corresponds to the parameter in `type =` that is in the stanzas in the **/etc/filesystems** file.

- To mount a file system that is not defined in the **/etc/filesystems** file, enter the following command:

  `mount –n hostname –v nfs [ –o options ] dirname1 dirname2`

  The *hostname* parameter identifies which server the file system is being mounted from.

  Note: If the first or second method is not chosen and the user wants the mount information to remain constant, then the **/etc/filesystems** may have to be changed to add an entry for an NFS mount.

Mount File Systems Using the automount Daemon

The **automount** daemon allows file systems to be mounted as needed. With this method of mounting directories, all of the file systems mounted with NFS do not need to be mounted all of the time. Only those being used will be mounted.

For example, to use the **automount** daemon to mount the `/usr/local/dir1` and `/usr/local/dir2` directories as needed from the `serve1` server onto the `/usr/remote/dir1` and `/usr/remote/dir2` directories respectively, perform the following tasks:

1. Create a file with the following contents. This file will be the map for the **automount** daemon. In this example, the map file name is `/tmp/mount.map`.

```
dir1            –rw             serve1:/usr/local/dir1
dir2            –rw             serve1:/usr/local/dir2
```

2. Verify that the directories on the server are correctly exported to the client.
3. Invoke the **automount** daemon, as follows:

```
/usr/etc/automount /usr/remote /tmp/mount.map
```

4. If the **automount** daemon is to be stopped, issue the kill command to the process ID of the **automount** daemon. This can be obtained by issuing the **mount** command with no parameters. The PID of the **automount** daemon will be displayed in the mounted column for the mount entry.

   Note: Do not use the **SIGKILL (kill –9)** signal or the **automount** daemon will not clean up before exiting.

   ```
   kill <automount_pid>
   ```

   The `<automount_pid>` is the ID of the daemon. This sends the **SIGTERM** signal to the **automount** daemon.

With this example, a user will be able to enter the command `cd /usr/remote/dir1` and have that directory mounted when the `cd` is done. With this method of mounting directories, all of the file systems mounted with NFS do not need to be mounted all of the time. Only those being used will be mounted.

Using NIS to Manage Maps Used by the automount Daemon

The **automount** daemon by default uses the NIS **auto.master** map. This NIS map has the following format:

```
Directory Path                  Automount Map Name
```

The **automount** daemon will look for this map if NIS is running. It reads the map to find those directories that it is supposed to be watching for mounts. The *AutomountMapName* parameter is a file name that contains the **automount** map for the corresponding directory. The **/etc/auto.master** file is usually where this special map resides on the NIS server. To construct this map, perform the following:

1. Create the **/etc/auto.master** file with the appropriate contents for the **automount** daemon.
2. Create the NIS map by modifying the **/etc/yp/Makefile** so that the **auto.master** NIS map will be built. The entry that needs to be added to the **Makefile** is the following:

```
auto.master.time: $(DIR)/auto.master
        -@if [ -f $(DIR)/auto.master ] ; then \
                $(MAKEDBM) $(DIR)/auto.master
$(YPDBDIR)/$(DOM)/auto.master; \
                touch auto.master.time ; \
                echo "updated auto.master" ; \
                if [ ! $(NOPUSH) ] ; then \
                        $(YPPUSH) auto.master ; \
                        echo "pushed auto.master" ; \
                else \
                        : ; \
                fi \
        else \
                echo "couldn't find $(DIR)/auto.master" ; \
        fi
```

In general, the same format that is used for the **netmasks** entry in the **Makefile** file can be used for the **auto.master** entry. After the **Makefile** has been updated, then the map can be built by executing the following command:

```
make auto.master
```

If errors are generated, they would probably be due to improper configuration of NIS, error in the **Makefile** or errors in the syntax of the **/etc/auto.master file**.

The contents of the **/etc/auto.master** file on the NIS server will look like the following:

```
/u/home         /etc/auto.home
/usr/lpp        /etc/auto.direct
```

This **auto.master** file will direct the **automount** daemon to use the automount map `/etc/auto.home` for the directory `/u/home` and the automount map `/etc/auto.direct` for the `/usr/lpp` directory. The contents of the `/etc/auto.home` file on the NIS client are:

```
john            -rw,hard,intr           host1:/u/john
bill            -rw,hard,intr           host3:/u/bill
sally           -rw,hard,intr           host5:/u/sally
fred            -rw,hard,intr           host9:/u/fred
jane            -rw,hard,intr           host1:/u/jane
```

The contents of the `/etc/auto.direct` file on the NIS client are:

```
X11             -ro,hard,intr           lppserver:/usr/lpp/X11
bsmEn_US        -ro,hard,intr           lppserver:/usr/lpp/bsmEn_US
gnuemacs        -ro,hard,intr           lppserver:/usr/lpp/gnuemacs
info            -ro,hard,intr           lppserver:/usr/lpp/info
```

Once the NIS client has the above two maps, and the **auto.master** NIS map is available from the NIS server, the **automount** daemon can be started. The syntax would be:

```
/usr/etc/automount
```

This starts the **automount** daemon, which reads the **auto.master** NIS map. It then obtains the necessary map information from the `/etc/auto.home` file and the `/etc/auto.direct` file.

When this process is completed, a user on the client will be able to issue the command `cd /u/home/bill` and have the `/u/bill` directory mounted from the `host3` system onto the `/u/home/bill` directory. The same is true if the **info** command is executed. The `/usr/lpp/info` directory is mounted from the `lppserver` system onto the `/usr/lpp/info` directory on the client and the **info** command is executed from the mounted directory.

Maintaining All of the automount Maps with NIS

In the example above, the `/etc/auto.home` and `/etc/auto.direct` were local files on the client that contained all of the **automount** map needed. The contents of the **automount** maps can also be maintained by NIS. The files would still exist on the client, but the contents would be different. For example, the `/etc/auto.home` file would contain the following:

```
+auto.home
```

And the **/etc/auto.direct** file would contain the following:

```
+auto.direct
```

This directs the **automount** daemon to consult the NIS maps `auto.home` and `auto.direct` when it reads the local files. The NIS server would contain two new NIS maps. The maps would be `auto.home` and `auto.direct`. They would be added to the **/etc/yp/Makefile** in the same way that the **auto.master** NIS map was added. This makes them available for use by the NIS clients running the **automount** daemon.

This facility can also be used to define local portions of the **automount** maps and then refer to the NIS maps for the rest of the **automount** map. For example, the `/etc/auto.home` file could contain the following:

```
sandy           -rw,hard,intr          host10:/u/sandy
james           -rw,hard,intr          host2:/u/james
bill            -rw,hard,intr          host20:/u/bill
+auto.home
```

This **automount** map has three local entries and then it contains the NIS map `auto.home`. This way, local definitions can be maintained while taking advantage of the NIS map for the `/u/home` directory. The entry `bill` in the local map would appear in the `auto.home` NIS map. The local map entry will override the NIS map entry.

Special automount Maps

There are two special maps that the **automount** daemon can use. They are described in the command documentation for **automount** daemon. The first is the `-passwd` map. It is not recommended that this special map be used. It does not work very well with the login process. The alternative is to use something like the `auto.home` map as described above. This can be developed to provide similar functionality. The other map is the `-hosts` map. This map works well and can be used to offer a powerful mounting tool for users. This map provides the means to mount all available directories from a particular server.

Related Information

The **chnfsmnt** command, **exportfs** command, **mknfsmnt** command, **mount** command, **rmnfsmnt** command, **showmount** command, **smit** command.

The **automount** daemon, **biod** daemon.

The **/etc/filesystems** file.

How to Use the NFS Secure Option When Mounting a Directory on page 11–85, How to Use the NFS Secure Option When Exporting a Directory on page 11–78, How to Unmount a File System Using NFS on page 11–87.

Administering Secure NFS on page 11–11.

How to Use the NFS Secure Option When Mounting a File System

Prerequisite Tasks or Conditions

1. Make sure that the system that is to be the server is currently exporting the directory that will be mounted on the local system. To do this, use the **showmount** command:

   ```
   showmount -e ServerName
   ```

   This shows the directories that are currently exported from the system specified by the *ServerName* parameter.

2. Make sure that a mount point exists on the client system for the file system that is to be mounted. The mount point may be created with the **mkdir** command.

3. Make sure that the **publickey** map exists and that the **keyserv** daemon is running. See Configuring Secure NFS Networking on page 11–16 for details on these prerequisites.

Procedure

The procedure for using the secure option to mount file systems is the same as the procedure for regular mounting except for a difference in choice of parameters. See How to Mount a File System Using NFS on page 11–80 for details on mounting, and add the steps below for using the secure option to mount file systems.

There are three ways to use the secure option, each of which corresponds to one of the three methods of mounting a file system:

1. If the SMIT interface is used (the **smit mknfsmnt** command) for mounting a directory, then the default of `no` should be change to `yes` for the selection `Use SECURE mount option?`.

2. If the **mknfsmnt** command is used, then the `-s` option should be used to specify that the secure option is to be used when mounting the specified directory.

3. If the **mount** command is used directly, then the `secure` option should be used as one of the options specified on the command line. For example:

   ```
   mount -o secure -n Server -v nfs /remote/dir /local/dir
   ```

Related Information

The **chnfsmnt** command, **exportfs** command, **mknfsmnt** command, **mount** command, **rmnfsmnt** command, **smit** command.

The **biod** daemon, **keyserv** daemon.

The **/etc/filesystems** file, **/etc/publickey** file.

How to Unmount a File System Using NFS on page 11–87, How to Mount a File System Using NFS on page 11–80.

Administering Secure NFS on page 11–11, Maintaining an NFS Client on page 11–34.

How to Unexport a Directory Using NFS

Prerequisite Tasks or Conditions

1. Install and configure TCP/IP.
2. Install NFS as described in How to Install AIX for RISC System/600 Licensed Program Optional Program Products in *General Concepts and Procedures.*
3. Make sure that NFS is currently running on your system. See How to Configure NFS on page 11–63 for details.

Procedure

This procedure describes how to unexport a directory that is currently exported using NFS. A directory may be unexported in one of the following ways.

1. Use the **smit rmnfsexp** command. This method uses the SMIT interface for unexporting directories with NFS.
2. Invoke the **rmnfsexp** command directly from the command line. The SMIT interface in the first method uses the **rmnfsexp** command to unexport the requested directory. If the user supplies the correct command line parameters, the **rmnfsexp** command may also be used from the command line.
3. Execute the **exportfs** command with the following format:

   ```
   exportfs -u /dirname
   ```

   where */dirname* is the full path name as specified in the **/etc/exports** file. If the export is to be removed permanently, edit the **/etc/exports** file and delete the entry for the directory.

Related Information

The **chnfsexp** command, **exportfs** command, **mknfsexp** command, **rmnfsexp** command, **smit** command.

The **nfsd** daemon, **mountd** daemon.

The **/etc/exports** file.

How to Create and Edit the NFS /etc/exports File on page 11–77, How to Use the NFS Secure Option When Exporting a Directory on page 11–78.

Administering Secure NFS on page 11–11, Maintaining an NFS Server on page 11–31.

How to Unmount a File System Using NFS

Prerequisite Tasks or Conditions

1. Make sure that the file system to be unmounted is currently mounted. To check the current list of mounted file systems, enter the **mount** command with no parameters.

Procedure

Unmount an NFS mounted directory in one of the following three ways:

2. Use the **smit rmnfsmnt** command. This method uses the SMIT interface for unmounting file systems from NFS Servers. Choose when the unmount procedure is to take place and execute the SMIT menu.
3. Invoke the **rmnfsmnt** command directly from the command line. The SMIT interface in the first method uses the **rmnfsmnt** command to unmount the requested file system. If invoked from the command line with the correct parameters, the **rmnfsmnt** command has the same effect as the SMIT interface.
4. Invoke the **umount** command at the command line. For example:

   ```
   umount /directory/to/unmount
   ```

   This unmounts the file system named `/directory/to/unmount`. If an entry exists in the **/etc/filesystems** file, that entry is not modified in any way. The first two methods can be used to remove an entry in the **/etc/filesystems** file that corresponds to an NFS mounted file system.

Related Information

The **chnfsmnt** command, **exportfs** command, **mknfsmnt** command, **showmount** command, **smit** command, **umount** command.

The **biod** daemon.

The **/etc/filesystems** file.

How to Use the NFS Secure Option When Mounting a Directory on page 11–85, How to Use the NFS Secure Option When Exporting a Directory on page 11–78, How to Mount a File System Using NFS on page 11–80.

Administering Secure NFS on page 11–11.

How To Edit the filesystems File

Prerequisite Tasks or Conditions

1. Install NFS as described in How to Install AIX for RISC System/600 Licensed Program Optional Program Products in *General Concepts and Procedures.*
2. Configure the server.

Procedure

This procedure shows how to establish the NFS default mounts by editing the **/etc/filesystems** file. Perform the following steps for each client in the network:

1. Sign on as root user by entering `root` at the login prompt. When the password prompt appears, enter your root password.
2. Using a text editor, open the **/etc/filesystems** file.
3. List the remote file systems that are to be mounted automatically when the system is booted.

   To do this, add a stanza to the file describing the file system you want to mount. The following attributes are *required* for the stanzas that pertain to the NFS mounts:

   `dev=`*`filesystem_name`*
   Specifies the path name of the remote file system being mounted.

   `mount=`*`false`* NFS mounts should use the *`true`* mount attribute for mounts that are mounted by default at system startup. For other mounts described by this file, make this value *`false`*.

   `nodename=`*`hostname`*
   Specifies the host machine on which the remote file system resides.

   `vfs=`*`nfs`* Specifies that the virtual file system being mounted is an NFS file system.

   The following attributes are *optional* in the stanzas for the NFS mounts:

   `type=`*`type_name`*
   Defines the file system being mounted as part of the *`type_name`* mount group. This parameter is used in conjunction with the **mount –t** command, which mounts groups of specified file systems at the same time.

`options=options`

Specifies one or more of the following `options`:

`biod` — Specifies the number of **biod** daemons to start. The default is 6.

`bg` — Specifies to try the mount again in the background if the first mount attempt fails.

`fg` — Specifies to try the mount again in the foreground if the first mount attempt fails.

The `bg` and `fg` options cause the **mount** command to run in the background (`bg`) or foreground (`fg`) if the server's **mountd** process does not respond. The system attempts to transmit each request the number of times specified in the `retry` option, before it gives up. Once the file system is mounted, each NFS mount request made in the kernel waits for a response during the time specified in the `timeout` option. If no response arrives, the timeout period is multiplied by two and the request is transmitted again. When the number of retransmitted attempts for a mount request that specifies the `soft` option have been sent with no reply, an error is returned. When the number of retransmitted attempts for a mount request that specifies the `hard` option have been sent with no reply, the system displays a message and retries the request.

`retry=n` — Sets the number of times to try the mount.

`rsize=n` — Sets the read buffer size to the number of bytes specified by *n*.

`wsize=n` — Sets the write buffer size to the number of bytes specified by *n*.

`timeo=n` — Sets the NFS timeout to the tenths of a second specified by *n*. Use this option to avoid situations that can occur in networks where the server load can cause inadequate response time.

`retrans=n` — Sets the number of NFS retransmissions to the number specified by *n*.

`port=n` — Sets the server port to the number specified by *n*.

`soft` — Returns an error if the server does not respond.

`hard`	Continues to try the request until the server responds. When you specify a `hard` mount, it is possible that the process can hang while waiting for a response. To be able to interrupt the process and end it from the keyboard, use the `intr` option in the mount options.
`intr`	Allows keyboard interrupts on hard mounts.
`ro`	Sets the read-only option.
`rw`	Sets the read-write option. Use the `hard` option along with this option to avoid error conditions that can conflict with applications if a `soft` mount is attempted as read-write. See Troubleshooting NFS-Related Problems on page 11–37 for information on hard- and soft-mounted problems.
`secure`	Specifies to use a more secure protocol for NFS transactions.
`actimeo=`*n*	Extends flush time by *n* seconds for both regular files and directories. The attribute cache retains file attributes on the client. Attributes for a file are assigned a time to be flushed. If the file is modified before the flush time, then the flush time is extended by the time since the previous modification (under the assumption that recently changed files are likely to change again soon). There are minimum and maximum flush time extensions for regular files and for directories.
`acregmin=`*n*	Holds cached attributes for at least *n* seconds after file modification.
`acregmax=`*n*	Holds cached attributes for no more than *n* seconds after file modification.
`acdirmin=`*n*	Holds cached attributes for at least *n* seconds after directory update.
`acdirmax=`*n*	Holds cached attributes for no more than *n* seconds after directory update.

Note: If you do not set the following options, the kernel automatically sets them to these defaults:

```
biod=6
fg
retry=10000
rsize=8192
wsize=8192
timeo=7
retrans=3
port=NFS_PORT
hard
secure=off
acregmin=3
acregmax=60
acdirmin=30
acdirmax=60
```

4. Remove any directory entries that you do not want to mount automatically at system startup.
5. Save and close the file.

Example

1. The following stanza can be created to set up an NFS client with a remote mount:

```
/u/jdoe:
      dev = /u/jdoe
      mount = false
      vfs = nfs
      nodename = mach2
      options = ro,soft
      type = nfs_mount
```

   This stanza directs the system to mount the `/u/jdoe` remote directory over the local mount point of the same name. The file system is mounted as read-only (`ro`). Since it is also mounted as `soft`, an error is returned in the event the server does not respond. By specifying the `type` as `nfs_mount`, the system attempts to mount `/u/jdoe` (along with any other file systems that are specified in the `type = nfs_mount` group) when the **mount –t nfs_mount** command is issued.

2. The following stanza can be created to set up an NFS client for an automatic remote mount:

```
/usr/games:
      dev = /usr/games
      mount = true
      vfs = nfs
      nodename = gameserver
      options = ro,soft,bg
      type = nfs_mount
```

   This stanza directs the system to mount the `/usr/games` file system at system startup time. If the mount fails, the system continues attempting to mount in the background.

Related Information

The **mount** command.

The **biod** daemon, **mountd** daemon.

The **/etc/filesystems** file.

How to Export a Directory Using NFS on page 11–79, How to Mount a File System Using NFS on page 11–80, How to Use the NFS Secure Option When Exporting a Directory on page 11–78, How to Configure an NFS Client on page 11–65.

Administering Secure NFS on page 11–11, Maintaining an NFS Client on page 11–34.

How to Add a New NIS Slave Server

Prerequisite Tasks or Conditions

NFS must be installed on your system.

Procedure

The procedure for modifying the **ypservers** map differs from other default maps because no text file is used as input for this map. Instead, the **makedbm** utility is used to create the modified **ypservers** maps. You must modify the **ypservers** map on the NIS master server.

Modifying the ypservers Map on the NIS Master Server

1. List the contents of the current **ypservers** map using the **makedbm** command with the **–u** flag. The output of this command, along with the echoed name of the new server, is then used as input to another **makedbm** command that creates the new map temporarily called `tmpsvrs`.
2. Change to the **/etc/yp** directory and enter the following:

   ```
   cd /etc/yp
   (makedbm –u domain/ypservers ; echo new_server) |
   makedbm – tmpsrvrs
   ```

   domain Specifies the name of the NIS domain (and the directory where the NIS maps are kept).

   new_server
   Specifies the name of the host being added to the **ypservers** map.
3. Check the contents of the new map by entering the following at the command line:

   ```
   makedbm –u tmpsrvrs
   ```
4. Use the **mv** (move) command to replace the old **ypservers** map files with the new ones. Enter the following at the command line:

   ```
   mv tmpsrvrs.pag domain/ypservers.pag
   mv tmpsrvrs.dir domain/ypservers.dir
   ```
5. Configure the new slave server.

Related Information

The **makedbm** command, **ypinit** command.

The **ypserv** daemon.

Network Information Service (NIS) Overview for System Management on page 11–47.

How to Add a New User to an NIS Server

Prerequisite Tasks or Conditions

1. NFS must be installed.
2. TCP/IP must be running.
3. The NIS master server must be configured.

Procedure

Add an entry to the password file.

1. Log in as root user on the NIS master server machine.
2. Add a new line to the password file by using the **mkuser** command on the NIS master server.
3. Update the NIS maps by changing directories to the **/etc/yp** directory and running the make command, as follows:

```
cd /etc/yp
make passwd
```

Related Information

The **mkuser** command, **yppasswd** command.

Network Information Service (NIS) Overview for System Management on page 11–47.

How to Change a New NIS Master Server

Prerequisite Tasks or Conditions

1. NFS must be installed on your systems.
2. TCP/IP must be running.
3. The master and slave servers must be configured.

Procedure

1. Build the new map on the new master server. It is not sufficient to use an existing copy of the map on the new master because the old NIS master server's name occurs as a key-value pair in the existing map. The key must be reassociated with the new master's name. If the map has an ASCII source file, this should be present in the map's current version on the new master.

   If the map exists only as a DBM database, you can remake the map on the new master by disassembling an existing copy (from any NIS server) with the **makedbm –u** command and then running the disassembled version back through the **makedbm** command.

2. Run the **ypxfr** command to retrieve a copy of the new map from the *new* master server. Have the old master then send copies of the new map to each of the slave servers because the slave servers will still have copies of the old map.

   Note: Do not use the **yppush** command first because the old master server does not have updated maps.

3. Run the **yppush** command to receive copies of the new map on the slave servers.

Related Information

The **makedbm** command, **yppush** command, **ypxfr** command.

Network Information Service (NIS) Overview for System Management on page 11–47.

How to Change an NIS Map

Prerequisite Tasks or Conditions

1. NFS must be installed on your system.
2. TCP/IP must be running.
3. The master server must be configured.

Procedure

In order to modify an NIS map, you need to be on the NIS master server.

Warning: Except for user password changes made with the **yppasswd** command, NIS maps should be modified on the NIS master server only. Modifying NIS maps on NIS slave servers can break the NIS service algorithm, which can result in unreliable NIS map data.

1. Edit the text file that is used as the input file for the map.
2. Change to the **/etc/yp** directory.
3. Run the **make** command to rebuild the new map by issuing the following at the command line:

   ```
   cd /etc/yp
   make map_type
   ```

   The *map_type* parameter specifies the NIS maps to be constructed from the input file.

Note: If you have modified several text files and want to confirm that all NIS maps are updated, issue the **make** command without parameters to automatically evaluate every input file on the NIS master server. If the file has been modified since the latest NIS map for that file was built, the NIS map is automatically rebuilt.

Related Information

The **make** command, **yppasswd** command.

Maintaining NIS on page 11–54.

Understanding NIS Maps on page 11–51.

How to Change NIS Passwords

Prerequisite Tasks or Conditions

1. NFS must be installed on your system.
2. The master server must be configured.

Procedure

Users can change their password by using the AIX **passwd** command if they are logged in on the NIS master server and the **/etc/passwd** file on the NIS master server is being used for NIS map input. If the **yppasswdd** daemon is running on the NIS master server, users may change their password from any host in the NIS domain using the **yppasswd** command.

The **yppasswdd** daemon, which runs on the NIS master server only, is usually started when the master server is started. See How to Configure the NIS Master Server on page 11–72. This daemon can also be started by using the **startsrc** command. See How to Start and Stop NIS Daemons on page 11–104.

By default, the **yppasswdd** daemon changes passwords for entries in the **/etc/passwd** file (the passwords are in the **/etc/security/passwd** file). To change the default password file to be updated, use the **chssys** command or the **startsrc** command with the **–a** option. For example, to change the **passwd** file into the **/etc/yp/passwd** file, enter:

```
startsrc –s yppasswd –a '/etc/yp/passwd –m passwd
       passwd=/etc/yp/passwd'
```

OR

```
chssys –s yppasswd –a '/etc/yp/passwd –m passwd
       DIR=/etc/yp'
```

Using the **startsrc** command causes temporary changes. Using the **chssys** command makes the change permanent.

Related Information

The **chssys** command, **make** command, **startsrc** command, **yppasswd** command.

The **yppasswdd** daemon.

Maintaining NIS on page 11–54.

How to Start and Stop NIS Daemons on page 11–104.

How to Create a New NIS Map from a Text File

Prerequisite Tasks or Conditions

1. Begin with a text file.

Procedure

In order to create an NIS map, you need to be on the NIS master server.

1. Modify the corresponding ASCII file on the master server so that the map can be rebuilt with the **make** command.

   Note: Consult the current **/etc/yp/Makefile** file to see examples of available utilities.

2. Modify the existing **/etc/yp/Makefile** file to contain any rules you wish to include. The new rules are processed when the **make** command is executed.

3. Run the **make** command, as follows:

   ```
   cd /etc/yp
   make
   ```

   Note: You can also rebuild the ASCII file by running the **makedbm** command from the command line to generate the NIS map.

4. Run the **yppush** command to propagate the map to all of the slave servers, or a single NIS slave server can retrieve the new map by issuing the **ypxfr** command.

The following example shows how to create a new map called `udir.nam` that lists the home directories of users in an NIS domain. The new map is created using filtered input from the NIS password file (**/etc/yp/passwd**). The keys for the map are the user names, and their corresponding values are the home directories. To begin creating the map, change to the **/etc/yp** directory and enter the following at the command line:

```
cd /etc/yp
awk '{FS=":" ; OFS="\t" ; print $1,$6}' /etc/yp/passwd | \

    makedbm - domain/udir.nam
```

The *domain* parameter specifies the current NIS domain.

Related Information

The **awk** command, **grep** command, **make** command, **makedbm** command, **sed** command, **yppush** command, **ypxfr** command.

Maintaining NIS on page 11–54.

Understanding NIS Maps on page 11–51.

How to Create a New NIS Map without a Text File

Prerequisite Tasks or Conditions

1. NFS must be installed on your system.

Procedure

In order to create a map you need to be on the NIS master server.

1. To use the **makedbm** command to convert input from standard input into an NIS map, enter:

```
cd /etc/yp
makedbm - home_domain/mymap
al ar
bl br
cl cr
<ctl D>
```

   Here the domain is `home_domain`.

2. Run the **yppush** command to propagate the map to all of the slave servers, or a single NIS slave server can retrieve the new map by issuing the **ypxfr** command.

Related Information

The **makedbm** command, **yppush** command, **ypxfr** command.

Understanding NIS Maps on page 11–51.

How to Update an NIS Slave Server Map

Prerequisite Tasks or Conditions

1. NFS must be installed on the systems.
2. The master server must be configured.
3. The slave server must be configured.

Procedure

Some maps require update verifications more frequently than others. For example, the **passwd** map can change many times a day and must be checked more frequently than the protocols or services maps, which may not change for months at a time. You can update maps periodically using the **cron** daemon.

To avoid having **crontab** entries for each map, you can group the **ypxfr** commands for several maps in a shell script and have the **cron** daemon execute the shell script at appropriate intervals.

1. Group the maps together according to how often they need updating. The following example shell scripts exist in the **/usr/etc/yp** directory:
   - **ypxfr_1perhour** (for checking each hour)
   - **ypxfr_1perday** (for checking once a day)
   - **ypxfr_2perday** (for checking twice a day).
2. Modify these shell scripts as necessary to meet the requirements of your site.
3. Use the **crontab** command to have the shell scripts executed automatically at specific intervals.
4. Add your new entries for the **ypxfr** procedures to the temporary file, using a text editor.
5. Modify the execution schedule for each entry according to how often each **ypxfr** script should be run.

Note: It is recommended that the request times differ for each NIS slave server to prevent overloading the NIS master server with requests from all NIS slave servers at once.

Although the **ypxfr** command can also be issued at the command line, you should only use it for special circumstances; for example, when creating a test NIS environment for bringing an NIS slave server into service after it has been down.

You can maintain a record of the **ypxfr** activity in a log file if you create the **/etc/yp/ypxfr.log**. To discontinue the **ypxfr** logging, remove the **/etc/yp/ypxfr.log** file.

Related Information

The **crontab** command, **ypxfr** command.

The **cron** daemon.

Maintaining NIS Servers on page 11–54.

Understanding NIS Maps on page 11–51.

How to Set Up NIS Client Files to Use NIS Services

Prerequisite Tasks or Conditions

1. NFS must be installed on your system.
2. The master server and slave servers must be configured.

Procedure

You should have all the machines on the NIS domain access the NIS maps on the NIS servers, rather than accessing potentially out-of-date information in their local administrative files. To enforce that policy, rename or edit many of the local administrative files in order to direct them to the NIS master server maps for current information. Perform the following tasks:

1. Run the **ypbind** daemon on the client machine.
2. Rename or edit the following files implemented by the NIS maps:
   a. Rename the **/etc/networks**, **/etc/protocols**, **/etc/ethers**, **/etc/services**, and **/etc/netgroup** files from the NIS clients. For example:

      ```
      mv /etc/networks /etc/networks-
      ```

   b. Add entries to the **/etc/hosts.equiv** file so that it references NIS. This reduces problems with any remote login or problems caused by different **/etc/hosts.equiv** files on the two machines.

      Edit the **/etc/hosts.equiv** file to contain a single line, with only the + (plus) character on it. This allows anyone to log on to the machine because all further entries are retrieved from NIS rather than the local file.

      For more control over logins, add a list of trusted hosts to the **/etc/hosts.equiv** file. For example:

      ```
      +@trusted_group1
      +@trusted_group2
      +@distrusted_group
      ```

      The names to the right of the `@` (at sign) should be netgroup names. The netgroup database is served by the NIS maps.

      Note: If none of the escape sequences are added to the **etc/hosts.equiv** file, NIS is not used.

   c. Add a list of trusted hosts to the **/.rhosts** file. For example:

      ```
      +@trusted_group1
      +@trusted_group2
      +@distrusted_group
      ```

      Because this file controls remote root access to the local machine, unrestricted access is not recommended. You cannot use secondary host names in the **.rhosts**, **hosts.equiv**, or **netgroup** files, because they all enable local machines to access remote machines. You can use secondary host names in the **/etc/hosts** file.

   d. Add entries for the local host name and the local loopback name to the **/etc/hosts** file. This file is accessed at boot time before NIS is available. After the system is running,

the **ypbind** daemon is started and the **/etc/hosts** file is not accessed again. The following is an example of the **hosts** file for a NIS client:

```
127.1          localhost
200.10.2.101   generic      #      your host name
```

e. Edit the **/etc/passwd** file to contain entries for the root user name and the primary users of the machine and add the NIS + (plus sign) escape entry to force the use of the NIS service. Also, add an entry for a daemon to allow file-transfer utilities to work. A sample of an NIS client **/etc/passwd** file follows:

```
root:!.k:0:1:/:/bin/csh
nobody:*:-2:-2::/:
daemon:*:1:1::/:
sys:*:2:2::/:/bin/csh
bin:*:3:3::/bin:
uucp:*:4:4::/usr/spool/uucppublic:
news:*:6:6::/usr/spool/news:/bin/csh
+::0:0:::
```

Note: The last line informs the library routines to use the NIS service. If you remove the last line in the **passwd** database, you will disable NIS password access.

A program that calls the **/etc/passwd** file first looks in the password file on your machine, then looks in the NIS password file if the client machine's password file contains + (plus) entries.

There are three types of + (plus) entries that may exist in this file. A + (plus) by itself means to include the entire contents of the NIS **passwd** map. A + (plus) with a name means to include that name from the NIS map. A `+@netgroup_name`, for example, means to insert the entries for all the members of the netgroup `netgroup_name` at that point. If the + (plus) entry contains data in one of the colon-separated fields of the password entry, that data overrides what is in the NIS map. This does not apply to the UID and GID fields. Also, earlier entries in the file take precedence over later entries with the same user name or user ID. Duplicate the entries for the **/etc/passwd** file and then duplicate the file on the other machines.

f. Reduce the **/etc/group** file to a single line to force translation of group names and group IDs through the NIS service. For example:

```
+:
```

Related Information

The **ypbind** daemon.

Changing Security with NIS on page 11–56.

How to Propagate an NIS Map

Prerequisite Tasks or Conditions

1. The NIS slave servers must be authorized to copy files remotely (using the **rcp** remote copy command) from the NIS master server.

Procedure

Copy updated maps from the master server using the **ypxfr** command in one of the following ways:

- Running the **cron** command periodically
- Running the **yppush** command from the master server
- Running the **ypxfr** command from the command line on the slave server.

Propagate Using the cron Daemon

1. Set up entries in the **crontab** file to run the **ypxfr** command as often as needed for any server in your NIS database.

   Note: The **ypxfr** command contacts the master server and transfers the database only if the master server's copy is more recent than the local copy.

2. Avoid having a **crontab** entry for each database by grouping frequent changes in a shell script and running a **/crontab** file. You can replace or modify the shell scripts (**ypxfr_1perhour**, **ypxfr_1perday**, **ypxfr_2perday**) that have been set up in the **/usr/etc/yp** directory.

   Note: Run the shell scripts from each NIS slave server. To avoid slowing the master server, alter the exact time of execution of the shell scripts on each server.

Propagate Using the yppush Command

1. Run the **yppush** command on the master server.

   The **yppush** command uses the **ypserver** database to generate a list of NIS servers in your domain. It then sends a *transfer database* request to the **ypserv** daemon on each of the slave servers.

   The **ypserv** daemon on the slave server executes a copy of the **ypxfr –C** command and then passes a summary of the information it needs to identify the database and call back the initiating **yppush** command.

Propagate Using the ypxfr Command on the Command Line

Only choose this option when setting up a temporary NIS server to create a test environment or when trying to bring up an NIS server that has been out of service.

1. Type **ypxfr** on the command line.
2. Transfers and attempts to transfer are logged in the **/etc/yp/ypxfr.log** file. If the file exists, results are appended to it. To turn off logging, remove the log file.

Related Information

The **cron** command, **rcp** command, **yppush** command, **ypxfr** command.

The **ypserv** daemon.

Understanding NIS Maps on page 11–51.

How to Start and Stop NIS Daemons

Prerequisite Tasks or Conditions

1. NFS must be installed on your system.
2. The **portmap** daemon must be running.

Procedure

NIS daemons are started and stopped by using the **startsrc** and **stopsrc** commands. The daemons are usually started by the commands used for configuring the NIS master server, slave servers, and clients. The daemons may be started and stopped individually by using the **startsrc** and **stopsrc** commands. The syntax is as follows:

```
startsrc -s daemon_name
```

```
stopsrc -s daemon_name
```

For example:

```
startsrc -s ypbind
```

```
stopsrc -s ypbind
```

To start or stop NIS as a group:

```
startsrc -g yp
stopsrc -g yp
```

The NIS daemons controlled by SRC are:

Daemon Name	Subsystem Name	Group Name
keyserv	keyserv	keyserv
ypbind	ypbind	yp
yppasswdd	yppasswdd	yp
ypserv	ypserv	yp
ypupdated	ypupdated	yp

Related Information

The **startsrc** command, **stopsrc** command, **ypinit** command.

The **ypbind** daemon, **yppasswdd** daemon, **ypserv** daemon, **ypupdated** daemon.

How to Configure the NIS Master Server on page 11–72, How to Configure NIS Slave Servers on page 11–73, How to Configure the NIS Clients on page 11–74.

System Resource Controller Overview in *General Concepts and Procedures*.

How to Edit the NIS Map Input Files

Prerequisite Tasks or Conditions

1. NFS must be installed on your system.

Procedure

Install user and group information in your NIS environment in one of the following manners:

- Use the /etc/passwd and /etc/group files on the NIS master server as the NIS map input files.
- Use separate password and group files as the NIS map input files.
- Do not use NIS for password and group information.

Determine which situation is appropriate for your NIS network environment by considering your administrative overhead.

Use the /etc/passwd and /etc/group Files as the NIS Map Input Files

NIS uses the **/etc/passwd** and **/etc/group** files from the NIS master server as input files for the NIS maps that contain password and group information.

All users and groups on the NIS master server are included automatically in the NIS maps. In a network that contains a number of machines, the user and group IDs supplied for a workstation may conflict with those on the NIS master server. This can cause file ownership problems if an NFS server exports a directory to an NFS client whose password file contains user IDs that match those in the NIS map.

Note: In order for users listed in the NIS password files to change their passwords, they must use the **yppasswd** command, and the **yppasswdd** daemon must be running on the NIS master server.

Use Separate Password and Group Files As NIS Map Input

To permit exclusion of selected entries from the NIS maps, you can create new files, such as **/etc/yp/passwd** and **/etc/yp/group**, that are for NIS users and groups only. Place entries designated for local use only in the **/etc/passwd** and **/etc/group** files. Place entries designated for NIS use in the newly created files. To configure the NIS master server in this way, do the following:

1. In the **/etc/passwd** file, verify that existing entries are local entries only and add the NIS escape sequence (`+::0:0::`) as the last line in the file.
2. In the **/etc/group** file, verify that existing entries are local entries only and add the NIS escape sequence (`+:`) as the last line in the file.

   Note: This is not the recommended method, since adding NIS escape sequences to the **/etc/passwd** and **/etcgroup** files on the master can create security holes.
3. Create the new files to be used instead of the **/etc/passwd** and **/etc/group** files. The new files should have the same format and name as the **/etc/passwd** and **/etc/group** files, but contain information for YP users only and reside in a different directory.
4. Modify the following variable in the **/etc/yp/Makefile**:
   change `DIR /etc` to `DIR=/new.dir`.

   Note: You cannot use AIX commands to create users and groups in files other than the **/etc/passwd** and **/etc/group** files.

Do Not Use NIS for Password and Group Information

In a typical NFS environment, home directories are exported over the network so that users can log in from any workstation and access their own working environment. To ensure proper permissions for remotely mounting file systems and to allow access to home directories throughout the network, NFS requires global user ID (UID) and group ID (GID) assignments.

If you choose not to use NIS to manage UID/GID assignments in an NFS environment, you must maintain duplicate password and group accounts for users who need to access multiple NFS hosts. For example, a user on a workstation whose UID is 205 and whose GID is 35 must have the same UID and GID (205 and 35) in the **/etc/passwd** file on every workstation that user accesses.

Note: If you intend to use secure RPC networking, you must use NIS to maintain the password and group information.

Related Information

The **yppasswd** command.

The **yppasswdd** daemon.

The **/etc/passwd** file, **/etc/group** file.

How to Configure the NIS Master Server on page 11–72, How to Start and Stop NIS Daemons on page 11–104.

Alphabetical List of NFS and NIS Daemons and Commands

The following alphabetical list shows the daemons and commands that are available for the NFS and NIS:

automount	Automatically mounts NFS file systems.
biod	Block input/output daemons that asynchronously process data contained in caches to facilitate the client's read and write requests to the server. This daemon is started from the **/etc/rc.nfs** file.
bootparamd	Provides information necessary for bootng to diskless clients.
chkey	Changes the user's encryption key.
chmaster	Re-executes the **ypinit** daemon and restarts the NIS daemons.
chnfs	Invokes a specified number of **biod** and **nfsd** daemons.
chnfsexp	Changes the attributes of an NFS-exported directory.
chnfsmnt	Changes the attributes of an NFS-mounted directory.
chslave	Retrieves maps from a master server and restarts the **ypserv** daemon.
chypdom	Changes the current domainname of the system.
domainname	Displays or sets the name of the current domain.
exportfs	Exports and unexports directories to NFS clients.
keyenvoy	An intermediary between user processes and the key server.
keylogin	Decrypts and stores the user's secret key.
keyserv	Stores public and private keys.
lockd	Processes lock requests through the RPC package.
lsmaster	Displays the characteristics of the configuration of an NIS master server.
lsnfsexp	Displays the characteristics of directories that are exported with NFS.
lsnfsmnt	Displays the characteristics of mounted NFS file systems.
makedbm	Makes an NIS map.
mkclient	Starts the **ypbind** daemons and uncomments the appropriate entries in the **/etc/rc.nfs** file.
mkkeyserv	Starts the **keyserv** daemon and uncomments the appropriate entries in the **/etc/rc.nfs** file.
mkmaster	Starts the NIS daemons on the master server.
mknfs	Configures the system to run NFS and starts NFS daemons.

mknfsexp	Exports a directory using NFS.
mknfsmnt	Mounts a directory using NFS.
mkslave	Retrieves maps from the NIS master server and starts the **ypserv** daemon.
mountd	Answers requests from clients for file system mounts.
newkey	Creates a new key in the **publickey** file.
nfsd	Starts the processes that handle a client's request for file system requests. This daemon is started from the **/etc/rc.nfs** file.
nfsstat	Displays information about a machine's ability to receive calls.
on	Executes commands on remote machines.
pcnfsd	Handles service requests from PC-NFS clients.
portmap	Maps RPC program numbers to port numbers. When a client makes a remote procedure call to a specific program, it contacts the **portmap** daemon, also known as the port mapper, to determine the port where the request should be sent. This daemon is started from the **/etc/rc.tcpip** file.
rexd	Executes programs on remote machines.
rmkeyserv	Stops the **keyserv** daemon and comments the entry for the **keyserv** daemon in the **/etc/rc.nfs** file.
rmnfs	Stops the NFS daemons.
rmnfsexp	Removes NFS-exported directories from a server's list of exports.
rmnfsmnt	Removes NFS-mounted file systems from a client's list of mounts.
rmyp	Removes the configuration for NIS.
rpcgen	Generates C code to implement an RPC protocol.
rpcinfo	Reports the status of RPC servers.
rstatd	Returns performance statistics obtained from the kernel.
rup	Shows the status of a remote host on the local network.
rusers	Reports a list of users logged on to the remote machines.
rusersd	Responds to queries from the **rusers** command.
rwall	Sends messages to all users on the network.
rwalld	Handles requests from the **rwall** command.
showmount	Displays a list of all clients that have mounted remote file systems.
spray	Sends a specified number of packets to a host.

sprayd	Receives packets sent by the **spray** command.
statd	Provides crash-and-recovery functions for the locking services on NFS.
ypbind	Enables client processes to bind, or connect, to NIS.
ypcat	Prints out the NIS map specified by the *MapName* parameter.
ypinit	Sets up NIS maps on an NIS server.
ypmatch	Displays the value of a given key within an NIS map.
yppasswd	Changes your network password in NIS.
yppasswdd	Receives and executes requests from the **yppasswd** command.
yppoll	Displays the order number (ID number) of the NIS map currently in use on the server.
yppush	Prompts the NIS slave servers to copy updated NIS maps.
ypserv	Looks up information in the local NIS maps.
ypset	Directs a client machine to a specific server.
ypupdated	Updates information in NIS maps.
ypwhich	Identifies either the NIS server or the server that is the master for a given map.
ypxfr	Transfers an NIS map from an NIS server to a local host.

Related Information

Functional List of NFS and NIS Daemons and Commands on page 11–110.

List of NFS Files on page 11–113.

Network File System (NFS) Overview for System Management.

Functional List of NFS and NIS Daemons and Commands

The following list shows the daemons and commands that are available for NFS and NIS, listed according to the following functions:

- NFS Daemons and Commands
- NFS Locking Daemons
- Network Service Daemons and Utilities
- Secure Networking Daemons and Commands
- NIS Daemons and Commands
- Sun Diskless Client Support.

NFS Daemons and Commands

automount	Automatically mounts NFS file systems.
biod	Sends the client's read and write requests to the server.
chnfs	Invokes a specified number of **biod** and **nfsd** daemons.
chnfsexp	Changes the attributes of an NFS-exported directory.
chnfsmnt	Changes the attributes of an NFS-mounted directory.
exportfs	Exports and unexports directories to NFS clients.
lsnfsexp	Displays the characteristics of directories that are exported with NFS.
lsnfsmnt	Displays the characteristics of mounted NFS file systems.
mknfs	Configures the system to run NFS and starts NFS daemons.
mknfsexp	Exports a directory using NFS.
mknfsmnt	Mounts a directory using NFS.
mountd	Answers requests from clients for file system mounts.
nfsd	Starts the daemons that handle a client's request for file system operations.
pcnfsd	Handles service requests from PC-NFS clients.
rmnfs	Stops the NFS daemons.
rmnfsexp	Removes NFS-exported directories from a server's list of exports.
rmnfsmnt	Removes NFS-mounted file systems from a client's list of mounts.

NFS Locking Daemons

lockd	Processes lock requests through the RPC package.
statd	Provides crash-and-recovery functions for the locking services on NFS.

Network Service Daemons and Utilities

nfsstat	Displays information about a machine's ability to receive calls.
on	Executes commands on remote machines.
portmap	Converts RPC program numbers into Internet port numbers.
rexd	Accepts request to run programs from remote machines.
rpcgen	Generates C code to implement an RPC protocol.
rpcinfo	Reports the status of RPC servers.
rstatd	Returns performance statistics obtained from the kernel.
rup	Shows the status of a remote host on the local network.
rusers	Reports a list of users logged on to the remote machines.
rusersd	Responds to queries from the **rusers** command.
rwall	Sends messages to all users on the network.
rwalld	Handles requests from the **rwall** command.
showmount	Displays a list of all clients that have mounted remote file systems.
spray	Sends a specified number of packets to a host.
sprayd	Receives packets sent by the **spray** command.

NFS Secure Networking Daemons and Utilities

chkey	Changes the user's encryption key.
keyenvoy	An intermediary between user processes and the key server.
keylogin	Decrypts and stores the user's secret key.
keyserv	Stores public and private keys.
mkkeyserv	Starts the **keyserv** daemon and uncomments the appropriate entries in the **/etc/rc.nfs** file.
newkey	Creates a new key in the **publickey** file.
rmkeyserv	Stops the **keyserv** daemon and comments the entry for the **keyserv** daemon in the **/etc/rc.nfs** file.
ypupdated	Updates information in NIS maps.

NIS Daemons and Commands

chmaster	Re-executes the **ypinit** daemon and restarts the NIS daemons.
chslave	Retrieves maps from a master server and restarts the **ypserv** daemon.
chypdom	Changes the current domainname of the system.
domainname	Displays or sets the name of the current domain.
lsmaster	Displays the characteristics of the configuration of an NIS master server.
makedbm	Makes an NIS map.
mkclient	Starts the **ypbind** daemons and uncomments the appropriate entries in the **/etc/rc.nfs** file.
mkmaster	Starts the NIS daemons on the master server.
mkslave	Retrieves maps from the NIS master server and starts the **ypserv** daemon.
rmyp	Removes the configuration for NIS.
ypbind	Enables client processes to bind, or connect, to NIS.
ypcat	Prints out the NIS map specified by the *MapName* parameter.
ypinit	Sets up NIS maps on an NIS server.
ypmatch	Displays the value of a given key within an NIS map.
yppasswd	Changes your network password in NIS.
yppasswdd	Receives and executes requests from the **yppasswd** command.
yppoll	Displays the order number (ID number) of the NIS map currently in use on the server.
yppush	Prompts the NIS slave servers to copy updated NIS maps.
ypserv	Looks up information in local NIS maps.
ypset	Directs a client machine to a specific server.
ypwhich	Identifies either the NIS server or the server that is the master for a given map.
ypxfr	Transfers an NIS map from an NIS server to a local host.

Sun Diskless Client Support

bootparamd	Provides information necessary for bootng to diskless clients.

Related Information

Alphabetical List of NFS and NIS Daemons and Commands on page 11–107.

List of NFS and NIS Files

The following list shows the file articles that are available for the Network File System (NFS):

bootparams	Lists clients that diskless clients can use for booting.
ethers	Lists Ethernet addresses of hosts on the network.
exports	Lists the directories that can be exported to NFS clients.
netgroup	Lists the groups of users on the network.
netmask	Lists network masks used to implement IP standard subnetting.
publickey	Stores public or secret keys from NIS maps.
rpc	Contains database information for RPC programs.
updaters	A makefile for updating NIS maps.
xtab	Lists directories that are currently exported.

Related Information

Alphabetical List of NFS and NIS Daemons and Commands on page 11–107.

Chapter 12. AIX Network Management/6000 Program

The AIX Network Management/6000 Licensed Program (Alerts Manager and the **xgmon** program) is a network management program for monitoring TCP/IP networks. It assists you in monitoring the status of the machines on a network and helps you anticipate network problems. This chapter contains information about the Alerts Manager application and daemons, and information about how to start, use, maintain, and customize the **xgmon** program.

Alert Manager Overview for Network Management

Alert Manager is a program written in C language that runs at the application level within the AIX operating system. It complies with the IBM OPEN Network Management approach and allows the AIX operating system and compatible applications to send *generic alerts,* in NMVT (Network Management Vector Transport) format, to the S/370 host NetView network management focal point program.

Alert Manager generates *generic alerts* to notify network operators of all significant network errors.

Alert Manager can be configured as an *entry point* or as a *collection point.*

Note: The terms *entry point* and *collection point* as defined and used by the Alert Manager program are product implemented; they are not functionally the same as the corresponding terms used by System Network Architecture (SNA) Management Services.

When configured as an *entry point,* the Alert Manager program sends alerts to an AIX Version 3 collection point node using Transmission Control Protocol/Internet Protocol (TCP/IP) or Systems Network Architecture (SNA) Control Point to Control Point (CP–CP) connection. If configured as a *collection point,* Alert Manager forwards generic alerts from its own node, or from other entry point nodes, to a focal point network management program such as NetView through the SNA System Services Control Point to Physical Unit (SSCP–PU) session.

Applications that run on AIX Version 3.1 can also log errors with the alert flag on in order to send alerts to the desired destination.

Notes:

1. The appropriate network session (for example, SNA SSCP–PU) must be activated before Alert Manager can send alerts to the focal point host. Alert transmission from an entry point node to a collection point node also requires the activation of an appropriate network session.
2. AIX Version 3.1 supports a SNA SSCP–PU session for a collection point node to send alerts to a focal point node. Both a SNA CP–CP connection and TCP/IP are supported for an entry point node to transmit alerts to a collection point node.

Generic Alerts

A generic alert contains the code points necessary to identify the following elements:

- A failure in the network (*Description of the Problem*)

- The causes of the failure in the network (*Probable Causes*)
- The actions necessary to correct (or further isolate) the failure in the network (*Recommended Action*).

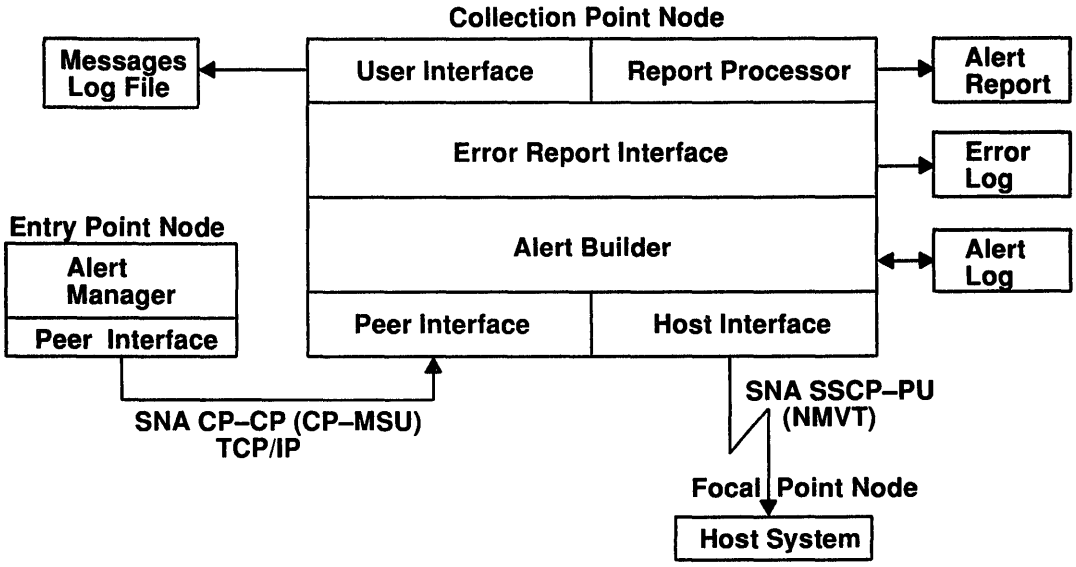
Dynamic Rerouting Feature in Alert Manager

Alert Manager provides a dynamic rerouting feature that automatically reroutes each alert if its primary routing configuration fails for any reason. Routing information from a secondary routing configuration is called upon to route the alert along an alternate path to a node where it can be examined and processed for problem resolution. Should the secondary routing configuration fail as well, a message to that effect is displayed on the screen and logged into the Alert Manager message file. At this point, all additional Alert Manager operations concerning the alerts depend upon whether the processing of held alerts has been specified in the primary profile.

During dynamic rerouting, the Alert Manager uses only the following **configam** command flag values from the secondary profile, ignoring all others:

–d *DestName*	The name of the destination
–n { **TCP/IP** | **SNA** }	The type of network being used to forward alerts
–p *APName*	The SNA attachment profile name. This flag is only valid when the **–n** flag is set to the **SNA** option and the **–t** flag is set to the **ep** (entry point) option. This flag is ignored for collection points.
–i *SSCPID*	The SSCP ID for the SNA connection to the host. This flag is only valid when the **–n** flag is set to the **SNA** option and the **–t** flag is set to the **cp** (collection point) option. This flag is ignored for entry points.

Note: The **–t** flag for the **configam** command specifies the type of configuration in use (entry point or collection point). It *must* be set to the same value in the secondary profile that it is in the primary profile. Otherwise Alert manager ignores both **–t** flag values completely.



Note: The internal structure of the Alert Manager configured as the Entry Point Node is not different from that of the Alert Manager configured as the Collection Point Node except that the Host Interface is not utilized.

Figure 1. Network Management Data Flow and Alert Manager Internal Structure

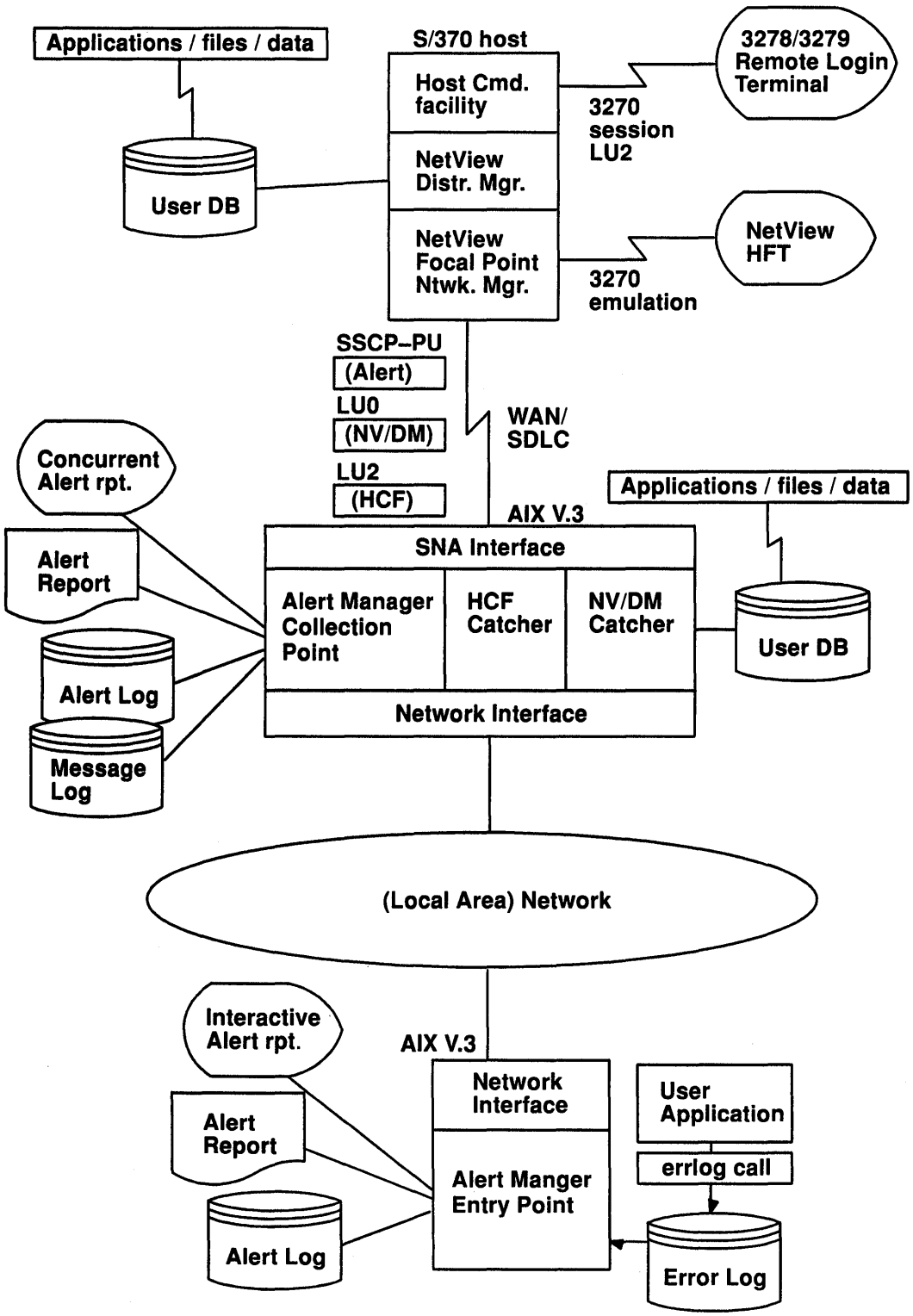


Figure 2. AIX Version 3 R1 Network Management Overview for SNA Network Configuration

How to Log Alertable Errors

Prerequisite Tasks or Conditions

None.

Procedure

1. Add an entry for each alertable error to the input file processed by the **errupdate** command:

   a. Assign a unique `#define` name for `LABEL`.

   b. Set the Alert flag to `TRUE`.

   c. Add a `Detail_Data` line for each subfield. Add as many as required by each code point for all code points that require the subfield.

   d. Find the correct `Data IDs` for the `Detail_Data` field.

   e. Add the following `Detail_Data` line for `Additional Subvectors` exactly as shown:

   ```
   Detail_Data = 208,8001,HEX
   ```

   Note: The Data ID of `8001` is reserved for the AIX operating system for processing the `Additional Subvectors` for alerts.

2. Call the **errlog** subroutine (by applications running above the kernel layer) or the **errsave** kernel service (by routines inside the kernel) to log errors as they occur.

   Notes:

   a. The `LABEL` is used for `error_id` when invoking the **errsave** kernel services because the actual value of the `error_id` field is unimportant.

   b. For errors that have recommended actions code points requiring the subfields, the `Detail_Data` field of the `err_rec` entry must have the following format:

   ```
   Data for subfields | As many subfield data as required |
   Additional Subvectors
   ```

3. Follow the procedures established for your Licensed Program Product (LPP) for dropping your Error Record Template descriptions into a build.

   Note: Error Record Template source files have a suffix of **.desc**. During the build process, before any other modules are compiled, the **errupdate** build tool is run to create the **.h** (header) file containing the #define names for the error ID's. The actual error IDs for all entries in the **.desc** are calculated at this time and placed in the header file with the define names taken from `LABEL` for each entry in the **.desc** file. The LPP may or may not ship this header file after compilation is completed.

4. You may need to run the **errupdate** command during the INSTALL or UPDATE process to process the **.desc** file and add entries to the Error Record Template Repository.

Related Information

The **configam** command, **controlam** command, **createoa** command, **rptalert** command, **sendtalrt** command, **startam** command, **stopam** command, **errupdate** command.

The **errlog** subroutine.

The **errsave** kernel service.

Introducing IBM AIX Systems Network Architecture Services/6000 on page 13–1 contains additional information about SNA Services.

The TCP/IP Overview for System Management on page 14–1 contains additional information about TCP/IP networks.

The Network Overview on page 5–1 contains additional information about the management and support of networks.

Host Command Facility Catcher and NetView DM Catcher Overview for Network Management

The following programs act as catchers (downstream servers), either for the S/370 Host Command Facility (HCF) program or for NetView Distribution Manager (NetView DM) program.

AIX Catcher (Downstream Server)	S/370 Host Application
hcfd	HCF
nvdmrelay	NetView DM.

Understanding the hcfd Catcher

The **hcfd** catcher (downstream server) program is designed to run transparently and continuously, much as a daemon does, either in an AIX shell or in the background. The program waits for a login request from the S/370 host HCF before initiating a specific *session* with the host HCF. You issue the following HCF commands to initiate and conclude each **hcfd** session from a 3270 terminal logged on to the host HCF:

Note: This program supports only Line Mode. Any command or program that invokes a full screen interface (such as the AIX **vi** editor or **smit**) is not supported.

- **ACQUIRE., *LUname*

  This command dispatches a bind to the **hcfd** program and performs other session initialization functions. When these initialization functions are completed, the AIX login prompt appears on the host terminal so that you can log in with your login ID and invoke AIX commands from the standard command line.

  Note: The *LUname* parameter is defined in the S/370 VTAM definition for communication from the S/370 host HCF program to the AIX **hcfd** program.

- **DROP.

  This command halts a session between the S/370 host HCF and the AIX **hcfd** program, leaving the host terminal logged onto the S/370 host HCF.

  Note: The host HCF checks for the **DROP command on a recurring basis, so you can enter the command at any time. However, the command *must* start in column 1 of the current line and be accompanied by at least one trailing space.

- **LOGOFF.

  This command halts a session between the S/370 host HCF and the AIX **hcfd** program, and then logs the host terminal off the HCF.

  Notes:

  1. To stop the **hcfd** program completely so that it does not continue to run on the AIX system, invoke the AIX **kill** command using the AIX process ID (PID) that is printed on the screen when the **hcfd** program is started. (This PID can also be obtained by invoking the AIX **ps** command.)
  2. The host HCF checks for the **DROP. and **LOGOFF, commands on a recurring basis, so you can invoke either command at any time. However, the command *must* start in column 1 of the current line and be accompanied by at least one trailing space.

An **hcfd** session begins when you log on to the host HCF in the following manner:

1. Invoke the HCF **ACQUIRE., *LUname* command from a 3270 terminal attached to a S/370 host.
2. Log onto the system through the standard AIX login prompt so that an AIX shell can be opened and AIX commands can be entered on the command line. (AIX commands that utilize curses or the graphics screen are not supported on the hcfd session).

   Note: The **hcfd** program uses pseudo terminal support to open the AIX shell required for the host session. For this reason, the pseudo terminal support must be available on your AIX system.
3. Issue the following command to allow some of the AIX screen positioning commands to work on the 3270 terminal (specifically those associated with Program Function (PF) keys PF1–PF12):

```
TERM=ibm3161;
export TERM
```

The two Systems Network Architecture (SNA) sessions associated with each **hcfd** session are described as follows:

- The SNA session between the host HCF and the downstream **hcfd** server is an LU0 LU to LU connection. It supports only the Line Mode.
- The SNA session between the host HCF and the host user terminal is an LU2 LU to LU connection.

You can conclude a **hcfd** session with any of the following commands:

- The AIX **kill** command (issued from a shell directly connected to the AIX RISC System/6000 node).
- The HCF **DROP. command
- The HCF **LOGOFF command
- **Ctrl–D** or PF22 (Must be issued from a 3270 terminal logged in to AIX.)
- The AIX **logout** command or PF24
- The AIX **exit** command
- The following special purpose PF keys implemented by the **hcfd** command:
  - PF22 = **Ctrl–D** (EOT)
  - PF23 = **CtrL–C**
  - PF24 = terminate **hcfd** session.

Understanding the nvdmrelay NetView DM Catcher

The **nvdmrelay** program acts as a catcher (downstream server) for the host NetView DM program or Distributed Systems Executive (DSX) program. It is designed to run transparently and continuously, interpreting commands for NetView DM and DSX. The **nvdmrelay** program supports the following NetView DM or DSX commands:

Command	Command Code
Delete Data	x'0283'
Execute Command List	x'0284'
Add Data with Resynchronization	x'8281'
Add Data without Resynchronization	x'0281'
Send Data with Resynchronization	x'8391'
Send Data without Resynchronization	x'0391'
Replace Data with Resynchronization	x'8282'
Replace Data without Resynchronization	x'0282'
Inform System Operator	x'0285'
Resync Send Data Response	x'8491'
Send Data Response	x'0491'
Status	x'05C0'

The **nvdmrelay** program opens the host connection using SNA Services and LU0 support, and issues a **read** so that it is ready to receive the first host command of the new session. As each host command is issued, **nvdmrelay** interprets it, processes it, and waits to receive the next host command or one of the following alternatives.

- The invoking of the AIX **kill** command (which stops the NetView DM catcher **nvdmrelay** program).
- The termination of the connection by the S/370 host (The session with the S/370 host NetView DM program is terminated and the NetView DM catcher waits until the next connection request by the S/370 NetView DM program.)
- The termination of program operation by an an uncorrectable error (such as an unrecoverable communications link failure).

Related Information

The **nvdmatoe** command, **nvdmconf** command, **nvdmconv** command, **nvdmetoa** command, **nvdmrelay** command, **hcfd** command.

Introducing IBM AIX Systems Network Architecture Services/6000 on page 13–1 provides detailed information about SNA Services.

Customizing AIX SNA/6000 on page 13–45 provides additional information about SNA/6000 and LU-to-LU connections.

The Alert Manager Overview for Network Management on page 12–1 contains additional information about the Alert Manager program.

The Network Overview on page 5–1 provides additional information about network management and support.

xgmon Overview for Network Management

The **xgmon** program is a network management program for monitoring TCP/IP networks. Through the use of SNMP (Simple Network Management Protocol), the **xgmon** program makes it possible for a network administrator to monitor the status of all of the machines on a network. The **xgmon** program helps the network administrator anticipate and correct network problems by communicating with SNMP agents and receiving SNMP-based traps.

This software is derived in part from the ISO Development Environment (ISODE). IBM acknowledges source author Marshall Rose and the following institutions for their role in its development: The Northrup Corporation and The Wollongong Group.

For more information on SNMP, you may want to read Understanding the Simple Network Management Protocol (SNMP) in *Communications Programming Concepts.* The following RFCs may also be of use:

- RFC 1098, Simple Network Management Protocol (SNMP)
- RFC 1066, Management Information Base for Network Management of TCP/IP-based internets (MIB)
- RFC 1065, Structure and Identification of Management Information for TCP/IP-based internets (SMI).

Related Information

How to Install the AIX Network Management/6000 Licensed Program in *General Concepts and Procedures.*

Enhanced XWindows Commands Overview in *General Concepts and Procedures.*

xgmon Overview for Programmers in *Communications Programming Concepts.*

Understanding the xgmon Program

The following sections provide general management information about the **xgmon** program and help you understand the working and use of the program.

You may also want to consult the **xgmon** Overview for Programmers in *Communications Programming Concepts.*

Getting Started

Before installing the **xgmon** program, install the X11 software, if desired, and install the TCP/IP software. To communicate with an SNMP agent on the client machine (that is, the machine on which **xgmon** will run), install the SNMP agent and start it up.

To set up the **xgmon** program on your network:

1. Install the **xgmon** program (see How to Install the AIX Network Management/6000 Licensed Program in *General Concepts and Procedures*).
2. Start the **xgmon** program (see page 12–23).
3. Create a topology description file (see page 12–25).
4. Create an **xgmon** startup file (**$HOME/.xgmonrc**) (see page 12–34).

Using an X11 Environment

The **xgmon** program can determine whether or not the X11 graphics interface is available. If X11 is available, the **xgmon** program can display multiple windows on a screen. Typically, one window serves as the console window, one displays a topology of the network, and the remaining windows display the output for the information entered in the console window. The output windows display the results of the processes running on the virtual G machines (VGMs). Multiple output windows allow the operator to inquire about the status of several machines at a time. Customize your X11 window by modifying the **$HOME/.Xdefaults** file. To do this, make appropriate changes to the sample **/usr/lpp/xgmon/smpl.Xdefaults** file, and append it to the end of the **$HOME/.Xdefaults** file.

In an X11 environment, it is possible to communicate information to a VGM using the topology display window and the mouse. Available to **xgmon** library programs are certain global variables that take on particular values when the mouse pointer is positioned on the topology display and the right button is clicked. See Working with the Topology Display Window on page 12–17 for more detailed information.

Using an ASCII Environment

If **xgmon** is invoked to run with the ASCII interface, graphics are not available. All executing virtual G machines will write their output to the standard output. Because graphics are not available, not all library programs will be able to successfully execute in the ASCII mode. The **xgmon** program is able to determine if a library program requires graphics capabilities. The **xgmon** program terminates all such executing library programs.

Version Control

If you encounter problems with the **xgmon** program itself, you can switch to an older or newer version of the program using runtime version control. When you invoke a different version of the **xgmon** program, the new version automatically accesses the corresponding startup script and library command directory as well. See Understanding Version Control on page 12–13 for more information.

Library and System Commands

There are two distinct classes of commands in the **xgmon** program:

- System commands

  System commands are built into the client (manager) and *cannot* be extended. They are generally used to control virtual G machines or the internals of the **xgmon** program, or both.

- Library commands

  Library commands are programs written in the **xgmon** programming utility, are stored as source code, and *can* be extended. The **xgmon** program provides a set of useful intrinsic functions that are used by library commands to perform the needed work. These intrinsic functions are of general utility and are not customized for any particular application.

  The **xgmon** program is actually a passive entity; it does not do anything on its own. All active processing is performed by library commands, which are external to the **xgmon** program. Any local customization of algorithms is embedded. The **xgmon** program is not normally modified, although it can be. The system programmer can tailor the **xgmon** program to fit the network's needs by adding new library commands or by modifying existing ones. See How to Create **xgmon** Library Commands and How to Modify Existing **xgmon** Library Commands in *Communications Programming Concepts* for more information.

  When a library command is issued, the **xgmon** program compiles it, loads it into the first available VGM, and executes it. When the **xgmon** program starts, it reads the current **xgmon** library directory, which is defined by the **GLIB** environment variable. This directory includes a set of the most commonly used library programs. However, the system programmer can write new library commands and add them to this directory. If library commands are added to this directory after the **xgmon** program is invoked, the library directory will automatically be reexamined when the new library command is referenced.

  New library commands can be added at any time and be used without having to terminate the current **xgmon** program. Similarly, bugs in library commands can be corrected without stopping everything. This means that algorithms can be corrected without losing global state information that has already been collected. Thus, the failure of one library command does not necessarily incapacitate the rest of the monitoring system. This feature makes the monitoring system as available as the network it is monitoring.

Understanding Version Control

If you encounter problems with a version of the **xgmon** software, you can terminate the currently executing **xgmon** program and switch to a previous or newer version. The **xgmon** client (manager) supports run-time version control, which permits the end user to select any of three distinct, potentially incompatible, **xgmon** releases. These releases are identified by their suffixes. To invoke one of these versions, specify which version you want as follows:

Name	Release Level
xgmon–	Previous release
xgmon	Current, production release
xgmon+	Newest, test release.

For more information, see How to Change to a Different Version of **xgmon** on page 12–31.

The network administrator is responsible for backing up all the **xgmon** files with the appropriate version control before installing a new release of the **xgmon** program. The **xgmon** installation procedure does *not* do this. The **xgmon** installation procedure *always* installs **xgmon** as the current production release (that is, without a suffix). The run-time control system does *not* enforce the meanings of different suffixes. It only provides for supporting distinct versions.

For more information on installing and invoking the **xgmon** program see How to Start **xgmon** on page 12–23, How to Install the AIX Network Management/6000 Licensed Program in *General Concepts and Procedures*, and the following section.

The runtime version control system determines which directories are used to store **xgmon** library commands as well as which startup script is used. Therefore, when you switch to a different version of the **xgmon** program, you change the program's default library directory and startup script as well. For instance, when you invoke the **xgmon** program with no suffix, the default library directory is **/usr/lpp/xgmon/lib**. Invoking the **xgmon–** version changes the default library directory to **/usr/lpp/xgmon/lib–**. Likewise, invoking the **xgmon+** version changes the default library directory to **/usr/lpp/xgmon/lib+**.

The **xgmon** client (manager) transparently handles any incompatibilities between versions, such as which levels of programming utilities are accepted, which library commands are used, or which startup commands are executed.

Note: You can override the **xgmon** library command directory name with the **GLIB** environment variable. In this case, when **xgmon** is invoked, the appropriate suffix is added to the name specified by the environment variable. The **GLIB** environment variable must be set *before* **xgmon** is invoked. Once **xgmon** is running, it is not possible to change the value of the **GLIB** environment variable.

Starting xgmon

When you invoke the **xgmon** program with the graphics interface, **xgmon** asks the X11 server indicated by the **DISPLAY** environment variable to create several windows for the output from the virtual G machines. As each window is displayed on the screen, the **xgmon** client (manager) decrements the number of windows and displays the new count.

After all of the windows are placed, the **.xgmonrc** file, found in the user's home directory, is read. If this file does not exist, it is not considered an error. The **xgmon** command you invoke determines which version of the startup file is used (see Understanding Version Control on page 12–13 for more information):

Command	File Version
xgmon–	**$HOME/.xgmonrc–**
xgmon	**$HOME/.xgmonrc**
xgmon+	**$HOME/.xgmonrc+**

For instance, if you invoke the **xgmon–** version of the **xgmon** program, the **$HOME/.xgmonrc–** file is started. Or, if you invoke the **xgmon+** version, the **$HOME/.xgmonrc+** file is started. Likewise, if you start the **xgmon** version (no suffix), the **$HOME/.xgmonrc** is started.

After processing the appropriate **.xgmonrc** file, the **xgmon** client (manager) prompts you for commands.

Note: It is possible to override the default **$HOME/.xgmonrc** startup file with a startup file with a different file name. See How to Start **xgmon** on page 12–23 and the **xgmon** command for more information.

Working with the Virtual G Machine (VGM) Output Windows

If you are using the X11 graphics interface with the **xgmon** program, you can display up to six output windows on the screen simultaneously. These windows are associated with virtual G machines (VGMs) and display the output produced by these VGMs.

The **xgmon** program contains a compiler that translates library programs into object code. This object code is interpreted by the VGMs. The **xgmon** program has a set of global variables that can be used to communicate information to a library program that is being executed by a VGM. Some of these global variables are set by mouse clicks.

A VGM executes the **xgmon** library program code. For instance, when you issue a library command, the virtual G machine executes the command and sends the output to the window associated with that VGM. The output can be in either text form or graphics form. The output can also be directed to a file instead of to a window. See How to Use Library Commands on page 12–38 and Programming Virtual G Machines (VGMs) in *Communications Programming Concepts* for more information.

Having several VGM output windows open at the same time makes it possible to receive output from several different programs simultaneously.

Notes:

1. Graphics are supported only when the **xgmon** program runs with the X11 graphics interface. Otherwise, ASCII terminals are supported. In this case, output from the VGM is not sent to a separate window but appears on the standard output instead.

2. If the graphics function is not available, some library commands may not execute.

Working with the Topology Display Window

When the X11 graphics interface is available, the topology display window displays a diagram, or topology, of the network. This diagram shows how the hosts and nodes in the network are linked to each other. The topology display supports a hierarchical organization. The layout of this topology is described in the current **xgmon** topology description. The current **xgmon** topology description is defined either by compiling a topology description file or by dynamically creating it. See How to Create a Topology Description File on page 12–25 and How to Update the Topology Description File on page 12–30 for more information on this file.

Each box in the diagram represents a host or node on the network. Each line represents a link between hosts or nodes. Each host, node, and link is a display element. A node has no IP address and is used to group together display elements (hosts or nodes, or both). A host has an IP address. Thus, it is not possible to link one node to another node. It is only possible to link to hosts. A link is a point-to-point link.

Each display element has a state associated with it. If you have a color monitor, these states are represented by different colors. If you have a monochrome monitor, the states are represented by various shades and labels.

The topology display is automatically updated when the **xgmon** program receives status information. Library programs executing in VGMs are not responsible for actively updating the status of displayed network elements. It is possible, if necessary, for a VGM to control how a given display element is colored.

To check the status of any *host* or *node* in the topology display window, compare its color to those in the following table:

Host or Node Color States		
State	**Color Monitor**	**B/W Monitor**
up/ok	green	white background
down	red	black background
unknown	yellow	shaded, white letters
acknowledge	cyan (blue-green)	shaded, black letters
ignore	violet	shaded, black letters
inactive	blue	shaded, black letters

To check the status of any *link* in the topology display window, compare its color to those in the following table:

Link Color States		
State	**Color Monitor**	**B/W Monitor**
up/ok	green line	solid line
down	red line	dotted line, large spaces
unknown	yellow line	dotted line, fine spaces
acknowledge	cyan (blue-green) line	thin dashed line
ignore	violet line	thin dashed line
inactive	blue line	thin dashed line

A display element that has a state of *down* or *unknown* generally indicates a problem. To acknowledge an element's state and to visually indicate the situation has been noticed, position the mouse pointer over the display element and press the middle button. This changes the element's state to *acknowledge.* When the element becomes operational again, the element's state automatically changes to *up.*

Occasionally, as when a network element is undergoing testing, you may want to ignore the state of an element. First, acknowledge it, then position the mouse pointer over the display element and press the middle button. This changes the element's state to *ignore.* To release the *ignore* state, position the mouse pointer over the element and press the middle button again. This changes the element's state to *unknown.*

The topology display supports a hierarchical organization. A user can view lower layers of the hierarchy by positioning the mouse pointer over a node and pressing the left mouse button. This creates a new window that displays the network elements grouped under that node. If the window already exists, a duplicate will not be created.

The topology display can be used to communicate information to a VGM. Global variables that can be set by mouse clicks are available to VGMs. See Working with Virtual G Machine (VGM) Variables in *Communications Programming Concepts* for more information.

For example, to communicate host or node name information to a VGM, position the mouse pointer over the host or node display element. Press the right button. This sets the name of the display element in the global variable *selected_host,* which can then be used by any library program that is coded to use it.

To associate a library command with a display element, issue the **setenv** library command as follows:

setenv *disp_elem* **RIGHTCLICK** *lib_cmd*

where

disp_elem Is the name of the display element with which you want to associate this library command. The name of is the name as displayed with the box on the topology display. If the display element is a link, the name can be found by issuing the query system command for one of the hosts at either end of the link.

lib_comd Is the name of the library command to be associated with this display element.

Note: The **RIGHTCLICK** parameter must be entered in all uppercase letters.

Once a display element is associated with a library command, the user can position the mouse pointer on the display element and double-click the right mouse button. This issues the library command.

The **setenv** intrinsic function can be issued within a library program to associate a library command with a display element. For example:

```
int return_code;
string disp_elem;
string lib_cmd;

disp.elem = ...;
lib_cmd = ...;
return_code = (int) setenv (disp_elem, RIGHTCLICK, lib_cmd);
```

For more information on how to use the VGM output windows see Working with the Virtual G Machine (VGM) Output Windows on page 12–16, the **setenv** subcommand, and the **setenv** intrinsic function.

You may also want to consult RFCs 1098, 1066, 1065.

Alphabetic List of Library Commands

bell	Controls the audible alarm.
cat	Concatenates the contents of a set of text files and writes the result to the output device of a virtual G machine (VGM).
clear_vgm	Clears the graphics window of the next available virtual G machine (VGM).
display	Specifies elements to display on the network topology display window.
dont_display	Specifies which elements not to display on the network topology display window.
dump_ipaddr	Displays the Internet Protocol (IP) address table of a Simple Network Management Protocol (SNMP) agent.
egp_chk	Checks the SNMP-based hosts for inactive or unknown EGP (Exterior Gateway Protocol) peers.
getenv	Displays the value of a user-defined environment variable for a display element.
load_env	Loads a predefined set of user-defined environment variables.
move	Changes the relative position of a display element within a topology display window.
perfmon	Displays a graphical representation of the number of incoming and outgoing packets on an SNMP agent.
ping	Checks whether a host on a network is responding to a **ping** query.
ping_all	Checks at a regular interval whether all the hosts defined in the current topology description respond to a **ping** query.
setenv	Sets a user-defined environment variable for a display element to a specified value.
snmp_get	Requests information from an SNMP agent.
snmp_next	Requests information from an SNMP agent.
snmp_p_all	Checks the status of interfaces on all SNMP-based agents defined in the current topology description.
snmp_p_list	Collects status information about SNMP agents defined in the current topology description.

snmp_set	Modifies values of read-write MIB variables managed by an SNMP agent.
trace_path	Determines the route between two hosts.
trap	Handles traps received by the **xgmon** program.

These library commands are described in *Commands Reference*.

Alphabetic List of System Commands

System commands are built into the **xgmon** program and control the virtual G machines (VGMs) in the following ways:

clearcache	Resets the internal cache of the host name/address database.
compile	Compiles a topology description file or an **xgmon** library program.
delay	Specifies the number of seconds the updating of the topology display window should lag behind in real time when negative state changes occur.
halt	Clears and unloads a virtual G machine.
help	Lists available commands.
hostdata	Specifies the auxiliary host name/address mapping file.
learn	Enables or disables the dynamic creation of display elements.
query	Queries the internal database for information about a host.
quit	Terminates the **xgmon** program.
readdir	Rereads the **xgmon** library command directory.
reuse	Enables or disables garbage collection.
start	Starts a loaded virtual G machine.
stat	Displays the status of virtual G machines currently in use.
stop	Suspends the execution of an **xgmon** library command.

These system commands are described in *Commands Reference*.

How to Start xgmon

Prerequisite Tasks or Conditions

1. Have root privileges or be in the system group to start the **xgmon** program.
2. Start up TCP/IP.
3. Start the X11 Windows program if you are using it.

Procedure

1. Change the current directory to **/usr/lpp/xgmon** by entering:

   ```
   cd /usr/lpp/xgmon
   ```

2. Set the shell environment variables as follows:

   a. Set the **PATH** environment variable by entering:

   ```
   PATH=/usr/lpp/xgmon/bin
   export PATH
   ```

   b. Set the **GLIB** environment variable to the directory containing the **xgmon** library commands by entering:

   ```
   GLIB=/usr/lpp/xgmon/lib
   export GLIB
   ```

   c. If you choose to change the **XGMONFONT** environment variable, select a font from the **/usr/lpp/fonts** file by entering, for example:

   ```
   XGMONFONT=8x13
   export XGMONFONT
   ```

   The **DISPLAY** environment variable must be set for the **xgmon** program to start up with the X11 interface. When X11 is invoked, the X11 **xinit** command automatically sets the **DISPLAY** environment variable to the specific X server on the system. Thus, the user need not set the **DISPLAY** environment variable. For further information, see Using Enhanced X-Windows for Display Specification in Enhanced X-Windows Commands Overview in *General Concepts and Procedures*.

   For a discussion of the shell environment variables, see **xgmon** Overview for Programmers in *Communications Programming Concepts*.

3. To start the **xgmon** program, enter the **xgmon** command. The **xgmon** command you invoke determines which version of the startup file is used:

Command	File Version
xgmon–	**$HOME/.xgmonrc–**
xgmon	**$HOME/.xgmonrc**
xgmon+	**$HOME/.xgmonrc+**

4. To override the default behavior of the **xgmon** program, invoke it with any of the following command line flags and parameters:

   - To start the **xgmon** program in the ASCII mode (without the X11 Windows program), add the **–nx** flag as follows:

     a. Make sure the enhanced editor is turned off by entering the following:

        ```
        stty –enhedit
        ```

     b. Enter:

        ```
        xgmon –nx
        ```

   - To set the number of virtual G machines (VGMs) you want, add the **–m** flag and specify the number of VGMs as follows:

     `xgmon –m` *`Number`*

     where *`Number`* is between 1 and 6.

   - To specify a startup file other than the default **$HOME/.xgmonrc** file, add the **–rc** flag and specify the file name as follows:

     `xgmon –rc` *`File`*

     Where *`File`* specifies the file name and path name of the startup file.

   - The **xgmon** program can also be passed one initial library command to be executed as a parameter on the command line when it is invoked. This permits library programs to be written and subsequently used somewhat like a shell script. When the **xgmon** program is invoked with an initial library command, it will automatically terminate when all of the virtual G machines become unloaded. The following example illustrates how this can be exploited:

     `xgmon –nx –rc /dev/null ping` *`HostName`* `> tmp`

     In this example, the **xgmon** command invokes the **xgmon** program and inhibits X11 (with the **–nx** flag). This command also overrides the default startup commands in the **$HOME/.xgmonrc** file by indicating that `/dev/null` is to be used instead (which will return end-of-file immediately). The **ping** library command is then run by the **xgmon** program. The shell has already taken care of redirecting the output into the `tmp` file. When the **ping** command terminates, the **xgmon** program will terminate automatically. The output from the command will be stored in the `tmp` file in the current library.

Related Information

The **xgmon** command.

How to Create a Startup File on page 12–34.

AIXwindows Window Management Overview, Enhanced X-Windows Commands Overview in *General Concepts and Procedures*.

How to Create a Topology Description File

Prerequisite Tasks or Conditions

1. Install the **xgmon** program.

Procedure

The topology description file defines the topology display window, including the nodes, hosts, interfaces, and links within it. The topology description file can be created in either of two ways:

- Manually at the keyboard
- Dynamically while running the **xgmon** program.

Note: To see a sample topology description file, go to the **/usr/lpp/xgmon/smpl.topology** file.

How to Create a Topology Description File at the Keyboard

Proceed as follows:

1. Go to the **/usr/lpp/xgmon** directory.
2. Name and open a file using a text editor.
3. Write your file according to the following template:

```
define window
  define nodes
  ...
  end nodes
  define interfaces
  ...
  end interfaces
  define links
  ...
  end links
end window
```

The `define window` and `end window` statements indicate to the **xgmon** compiler that the window definition you are creating is a topology description file.

As shown in the template, the window definition consists of the three following definition sections each defined by the appropriate `define` and `end` statements:

- Host and node definition section
- Interface section
- Link definition section.

The sequence of the definition sections should be as outlined above.

4. To include another topology description file, split up that file into three pieces corresponding to the three above-mentioned definition sections, and include each piece in the appropriate section using the `include` statement:

   ```
   include "File"
   ```

   where *File* specifies the file to be included.

   Note: The include directive must begin in the first column of the line if it is to be recognized.

Host and Node Definition Section

Write the host and node definition section according to the following template:

```
define nodes
  display...
  use...
  group...
end nodes
```

Then, proceed as follows:

1. Use the `display` statement to specify where a node or host must be displayed in the topology display window. Use one of the following statements:

   ```
   display node NodeName at x,y ("State");
   ```

   ```
   display NodeName at x,y;
   ```

   ```
   display HostName at x,y;
   ```

   where:

NodeName	Is the name of a node.
HostName	Is the name or IP address of a host.
x,y	Are the coordinates where the upper left corner of the display element representing this node or host should be located in the topology display window. The coordinates are relative to a 100 by 100 grid. The actual placement of an object is determined by scaling the specified coordinates relative to the actual size of the topology display window.
State	Indicates the initial state for this display element. The default is **unknown**. Any valid state as defined in Working with the Topology Display Window can be specified.

   Notes:

   1. The **node** keyword is optional. If this keyword is used, *no* IP address should map to the node name.
   2. If a host is specified by a name, the name must be able to be mapped to an IP address that is found in the **etc/hosts** file or that is known by the local name server. If the host name cannot be mapped, it is assumed that it is the name of a node.
   3. A box will be drawn large enough to include the name of the host or node. A box representing a node will be drawn somewhat larger than a box representing a host.

4. Node names and host names are case-insensitive. Thus, no two display elements can have the same name. For example, `Austin` and `austin` denote the same name.

2. Use the `use` statement to specify that the Simple Network Management Protocol (SNMP) is supported by a specific host:

   ```
   use snmp with HostName (Password);
   ```

   where:

   HostName Is the name or IP address of a host (SNMP agent).

   Password Is the community name of the SNMP agent.

   Note: The `use` statement cannot precede the `display` statement of the host it references. As a matter of style, the `use` statement should immediately follow the `display` statement of the host for which it is specifying the network management protocol to be used.

3. Use the `group` statement to group one or more hosts under a node. It is also possible to group one or more minor nodes under a major node. This permits the creation of a hierarchy of topology windows.

   ```
   group {HostName1, ..., HostNamen} under NodeName;
   ```

   where:

   HostName Is the name or IP address of a host.

   NodeName Is the node under which these hosts are to be added.

   Notes:

   1. The `group` statement must follow all the display elements of the host(s) and the node that are listed in the group statement.
   2. Multiple group statements can be used to group additional hosts or nodes under a node. Each group statement adds the specified set of display elements to the set that is already grouped under the node.
   3. At a top-level window, only the node will be displayed. A click with the left mouse button will pop up the lower-level topology display window and display the group of hosts or nodes, or both, listed in the group statement.

Interface Section

Write the interface definition section according to the following template:

```
define interfaces
  ... interfaces are ...
end interfaces
```

If there are no interfaces to be defined, type:

```
no interfaces
```

Use the `interfaces` statement to specify the interfaces for each host:

```
HostName interfaces are {Iface1, ... , Ifacen}
         alternate interfaces are {Iface2, ... , Ifacen};
```

where:

HostName Is the name of a host that was previously described by a display statement.

Iface Is an interface of a host specified either as a host name or as an IP address in dot notation.

Notes:

1. A host has often only one host name but multiple interfaces. (The host name then normally represents the primary interface to this host.) You can use the host name as *Iface1*, and list the other interfaces in dot notation.
2. Some implementations of SNMP agents will only accept SNMP requests if the interface they were sent to is operational; otherwise they will discard the request. This may happen if a request is sent to *Iface1* while *Iface1* is down. Since there are multiple interfaces, the request may still arrive at the host via one of the other interfaces. The optional alternate interfaces definition allows library programs to switch to the next alternate interface address in case they find that the primary interface is failing.

Link Definition Section

Write the link definition section according to the following template:

```
define links

  link...

end links
```

If there are no links to be defined, type:

```
no links
```

Use the `link` statement to specify the link between two hosts:

```
logical link from Iface1 to Iface2;
```

where:

Iface Is the name of a host or an IP address in dot notation corresponding to the IP address of each end of a point-to-point link.

Notes:

1. Logical links are considered to be point-to-point links.
2. The **logical** keyword is optional and may be omitted.

How to Create a Topology Description File Dynamically

To create a topology description file dynamically, you must first make sure that the **xgmon** program is running.

You can start with a minimally defined topology description file and create a fully defined topology description from the **xgmon** console dynamically. The shipped **/usr/lpp/xgmon/smpl.emptytdf** sample file is such a minimally defined topology description file.

This **/usr/lpp/xgmon/smpl.emptytdf** sample file contains one display element. Because there is no IP address defined for this display element, it will be treated as a node by the **xgmon** program. If you wish for it to be treated as a host, edit the file and change the host name to be that of your **xgmon** client machine. In addition, alter the interface section to include the IP address or addresses of your host.

To create the topology description file dynamically, proceed as follows:

1. Enter the **learn** system command as follows:

   ```
   learn on
   ```

2. Create nodes, hosts, and links dynamically in either of two ways:

   - Use the **mkdep** and **mklink** library commands. Note that these **xgmon** library programs are provided as sample programs for your use and are not documented.
   - Write your own library programs that make calls to the following **xgmon** intrinsic functions:

     – **group_dep**
     – **make_dep**
     – **make_link**
     – **move_dep**
     – **new_deps**

   As new nodes, hosts, and links are dynamically created, they become defined in the current topology description.

3. Use the **save_win** library command to save the topology description to a file. Note that this **xgmon** library program is provided as a sample program for your use and is not documented. Type:

   ```
   save_win File
   ```

   where *File* is the name of the file into which the topology description will be written.

Related Information

The **learn** system command.

The **group_dep** function, **make_dep** function, **make_link** function, **move_dep** function, **new_deps** function.

How to Update the Topology Description File on page 12–30.

Working with the Topology Display Window on page 12–17.

How to Update the Topology Description File

Prerequisite Tasks or Conditions

1. Make sure the **xgmon** program is installed.

Procedure

To set up the way you want the topology display window to look, or to add information about any of its display elements, edit the topology description file. Through this file you can modify the characteristics of the topology display window or of any of its display elements. Some of the modifications you can make are:

- Adding or deleting display elements.
- Adding or changing the characteristics of a display element such as IP address, interface, or link.
- Changing the x and y locations of a display element.
- Moving a node to a subnet.

You can modify the topology description file either manually or dynamically.

To modify the topology description file manually, proceed as follows:

1. Go to the **/usr/lpp/xgmon** directory by typing:

   ```
   cd /usr/lpp/xgmon
   ```

2. Using a text editor, open the topology description file. For instance, if you are using the vi text editor, type:

   ```
   vi File
   ```

3. Add or change entries for the characteristics you want to add or change. Save your changes.
4. Go back to the shell in which the **xgmon** program is running.
5. Enter the **clearcache** command. This updates **xgmon**'s internal cache.
6. Recompile the topology description file by invoking the **compile** command with this file.

You can dynamically modify the topology description file by dynamically creating new nodes, hosts, and links, and saving the file with the **save_win** library command, as described under How to Create a Topology File Dynamically in How to Create a Topology Description File.

Related Information

The **clearcache** system command, **compile** system command.

How to Create a Topology Description File on page 12–25.

Working with the Topology Display Window on page 12–17.

How to Change to a Different Version of xgmon

Prerequisite Tasks or Conditions

1. Have the **xgmon** program running.

Procedure

If you encounter problems with a version of the **xgmon** software, you can switch to a previous or newer version. To invoke a different **xgmon** version, proceed as follows:

1. Terminate the program.
2. Invoke the version you want as follows:

Name	Release Level
xgmon–	Previous release
xgmon	Current, production release
xgmon+	Newest, test release.

For example, if you are using the **xgmon+** version and you wish to switch to an older version, terminate the program and invoke **xgmon** (no suffix). Likewise, if you are using the **xgmon** (no suffix) version and you want to switch to an even older version, invoke **xgmon–**.

Related Information

How to Start **xgmon** on page 12–23.

How to Customize the X11 Environment for xgmon

Prerequisite Tasks or Conditions

1. Start the X11 Windows program if you are using it.

Procedure

The **xgmon** program uses the **.Xdefaults** file as initialization to find your preferences for window defaults. To tailor the appearance of the X11 screen to meet your system management needs, edit the **.Xdefaults** file. To see a sample **.Xdefaults** file, go to the **/usr/lpp/xgmon/smpl.Xdefaults** file. You can append this file to the bottom of your **$HOME/.Xdefaults** file. Through this file, you can specify the following characteristics for the X11 window:

- Placement of the topology window and subnet topology windows
- Placement of the VGM output windows
- Colors of the display elements
- Monochrome or color mode
- Color of the background or foreground
- Color of the cursor
- Color of the borders.

The **xgmon** program recognizes the standard keyword geometry; it recognizes:

- The following **xgmon** specific class identifiers:

  Top Topology windows

  G Virtual G machine windows

  Color Colors associated with states of display elements.

  Within the **Color** class of resources, the following keywords apply:

  down Specifies color for the down state.

  up Specifies color for the up state.

  inactive Specifies color for the inactive state.

  acknowledge Specifies color for the acknowledge state.

  unknown Specifies color for the unknown state.

- The following **xgmon** specific name identifiers:

  autoplace Specifies (yes/no) if automatic placement of windows is wanted. If you specify a location (x and y coordinates) in your geometry specification, this is considered to be an implicit yes for the autoplace resource. If you do *not* specify a location but *do* specify that autoplace is yes, the **xgmon** programwill place windows automatically at an internally defined location.

  force_monochrome Specifies (yes/no) if you want to force monochrome mode on a color display.

font Specifies the font to be used for text. The font specified should be of fixed width for proper orientation.

XGMON_SNMP_Manager
Specifies the name of the top-level topology display. This name is only an example of the name of a topology display. You can use any node name you specify on a display statement in the topology description file. You must use underscores to represent blanks.

To edit the **.Xdefaults** file:

1. Go to your home directory and, using a text editor, open the **.Xdefaults** file.
2. Modify the **.Xdefaults** file by adding or changing an entry for each characteristic you want to add or change.
3. To activate the changes, restart the X11 Windows program.

Related Information

How to Create a Startup File on page 12–34, How to Create a Topology Description File on page 12–25, How to Start **xgmon** on page 12–23.

How to Install the AIX Network Management/6000 Licensed Program in *General Concepts and Procedures.*

Enhanced X-Windows Commands Overview in *General Concepts and Procedures.*

How to Create a Startup File

Prerequisite Tasks or Conditions

1. Make sure the **xgmon** program is installed on your machine.

Procedure

1. Go to your home directory by typing:

   ```
   cd $HOME
   ```

2. Using a text editor, open the **.xgmonrc** file. For instance, if you are using the vi text editor open the file as follows:

   ```
   vi File
   ```

   where the *File* parameter specifies a name for the **.xgmonrc** file. By default, the **xgmon** program searches for the **.xgmonrc** file. You can either name the file **.xgmonrc** or use another name of your choice.

3. Add comment lines to the file if desired. Comments begin with the # (pound sign) in column one.

4. Add the commands that should execute when the **xgmon** program starts.

   Library commands you may want to include are:

   - **load_env** *EnvFile*

     where *EnvFile* is the name of the file containing information for the environment to be defined.

   - **ping_all**

   - **snmp_p_all**

   You may also want to add the following system command:

   - **compile** *TopFile*

     where *TopFile* is the name of the topology description file.

   Note: A maximum of six VGMs can be created. If you do not have enough VGMs available for the commands you add, you will get error messages when you start the **xgmon** program, and some library commands will not execute.

Related Information

The **compile** subcommand, **load_env** subcommand, **ping_all** subcommand, **snmp_p_all** subcommand.

How to Create a Topology Description File on page 12–25, How to Create Different Versions of the Startup File on page 12–35, How to Customize the X11 Environment for **xgmon** on page 12–32, How to Start **xgmon** on page 12–23.

How to Install the AIX Network Management/6000 Licensed Program in *General Concepts and Procedures.*

How to Create Different Versions of the Startup File

Prerequisite Tasks or Conditions

None.

Procedure

1. Go to your home directory by typing:

   ```
   cd $HOME
   ```

2. Using a text editor, open the **.xgmonrc** file. For example, if you are using the vi text editor, open the file as follows:

   ```
   vi .xgmonrc
   ```

3. Add or delete the library commands that should execute when the **xgmon** program starts.

4. Save the file to another file name. For example, if you are using the vi text editor, save the file as follows:

   ```
   :w! File
   ```

5. Exit the open **.xgmonrc** file *without* saving your changes.

Related Information

How to Create a Startup File on page 12–34, How to Create a Topology Description File on page 12–25, How to Customize the X11 Environment for **xgmon** on page 12–32, How to Start **xgmon** on page 12–23.

How to Install the AIX Network Management/6000 Licensed Program in *General Concepts and Procedures*.

How to Rename the Startup File

Prerequisite Tasks or Conditions

1. Make sure the **xgmon** program is installed on your machine.

Procedure

1. Go to your home directory by typing:

   ```
   cd $HOME
   ```

2. Using a text editor, rename the **.xgmonrc** file. For example, if you are using the vi text editor, type:

   ```
   mv .xgmonrc MyStartUpFile
   ```

Related Information

The **xgmon** command.

How to Create a Startup File on page 12–34, How to Start **xgmon** on page 12–23.

How to Install the AIX Network Management/6000 Licensed Program in *General Concepts and Procedures*.

How to Use System Commands

Prerequisite Tasks or Conditions

1. Have the **xgmon** program running.
2. If you want to see a list of system commands and library commands, type `help` and press Enter.

Procedure

Enter on the **xgmon** command line the name and any required parameters of the system command you want to invoke. Press Enter.

Related Information

How to Use Library Commands on page 12–38.

How to Use Library Commands

Prerequisite Tasks or Conditions

1. Have the **xgmon** program running.
2. If you want to see a list of system commands and library commands, type `help` and press Enter.

Procedure

Enter on the **xgmon** command line the name and any required parameters of the library command you want to invoke. Press Enter.

Note that library commands are invoked similar to AIX commands. A library command invocation has the following format:

Command Parameter1...Parametern > File

A library command may or may not take parameters. The output from a library command can be redirected into a file instead of being sent to the virtual G machine standard output device. If the path name of the output file is not specified, the file is written to the current directory.

In the following example, the output from the **ping** library command is redirected to a file:

`ping` *`HostName`* `>` *`File`*

where *`HostName`* is the name or Internet Protocol address of the host to be queried by the **ping** library command, and *`File`* is the name of the file into which the output will be redirected.

Related Information

How to Use System Commands on page 12–37.

Index

Numbers

A

B

C

D

E

F

G

H

I

K

L

M

N

O

P

Q

R

S

T

U

V

X

Y

Reader's Comment Form

AIX Communication Concepts and Procedures for IBM RISC System/6000

SC23–2203–00

Please use this form only to identify publication errors or to request changes in publications. Your comments assist us in improving our publications. Direct any requests for additional publications, technical questions about IBM systems, changes in IBM programming support, and so on, to your IBM representative or to your IBM-approved remarketer. You may use this form to communicate your comments about this publication, its organization, or subject matter, with the understanding that IBM may use or distribute whatever information you supply in any way it believes appropriate without incurring any obligation to you.

☐ If your comment does not need a reply (for example, pointing out a typing error), check this box and do not include your name and address below. If your comment is applicable, we will include it in the next revision of the manual.

☐ If you would like a reply, check this box. Be sure to print your name and address below.

Page	Comments

Please contact your IBM representative or your IBM–approved remarketer to request additional publications.

Please print

Date _____

Your Name _____

Company Name _____

Mailing Address _____

_____

_____

Phone No. ( ) _____
Area Code

No postage necessary if mailed in the U.S.A

NO POSTAGE
NECESSARY
IF MAILED
IN THE
UNITED STATES

BUSINESS REPLY MAIL

FIRST CLASS PERMIT NO. 40 ARMONK, NEW YORK

POSTAGE WILL BE PAID BY ADDRESS-
EE

International Business Machines Corporation
Department 997, Building 997
11400 Burnet Rd.
Austin, Texas 78758–3493

Fold

Fold

Fold and Tape

Please Do Not Staple

Fold and Tape

International Business Machines
Corporation
11400 Burnet Road
Austin, Texas 78758–3493

Printed in the
United States of America

SC23-2203-00

SC23-2203-00