ANNEX
COMMUNICATIONS SERVER

Network Administrator's Guide

Xylogics

166-024-000
Revision E
November 1991

ANNEX COMMUNICATIONS SERVER

Network Administrator's Guide

166-024-000
Revision E
November 1991

Notice

The information in this manual is subject to change without notice, and should not be construed as a commitment by Xylogics, Inc. Xylogics assumes no responsibility for any errors that may appear in this document.

Printed in the USA.

Contents

Book B: Configuration Procedures

Book C: Network Management

Book D: Reference

List of Figures

List of Tables

Preface

General

This guide is intended for the person responsible for installation, configuration, and day-to-day administration of the Annex Communications Server. The Annex operates in heterogeneous network environments. It can communicate with any system that supports the TCP/IP and LAT protocols. The Annex TCP/IP implemention is derived from the 4.3BSD tahoe distribution of UNIX, as are the implementations of several higher-level Internet protocols.

This guide assumes its readers have a basic familiarity with system and network administration in general, with the host operating system, and with the serial devices connected to the Annex. The host operating systems referenced are versions of UNIX distributed by the University of California at Berkeley, 4.2BSD and 4.3BSD, or versions of System V. The guide is organized into four books:

- *Book A: Overview* presents an introduction to the Annex, its features, its network applications, and the network protocols it supports.
- *Book B: Configuration Procedures* provides instructions on configuring the Annex, the serial ports, the parallel port, and any required service. Each chapter within this book covers a specific area or task.
- *Book C: Network Management* provides information on managing the network using tools and utilities supplied with the Annex.
- *Book D: Reference* provides reference material on the Annex tools and utilities, including the network administrator (**na**) utility and the Command Line Interpreter (CLI).

Supported Version

This guide supports R6.1 of the Annex Communications Server software.

Printing Conventions

This manual uses the following printing conventions:

`special type`	In examples, special type indicates system output.
`highlight`	Highlighted special type indicates user input.
RET	In command examples, this notation indicates that pressing the **Return** key enters the default value.
lowercase bold	Lowercase bold indicates commands, pathnames, or filenames that must be entered as displayed.
lowercase italics	In the context of commands and command syntax, lowercase italics indicate variables for which the user supplies a value.
[ ]	In command dialogue, square brackets indicate default values. Pressing the **Return** key selects this value. Square brackets appearing in command syntax indicate optional arguments.
{ }	In command syntax, braces indicate that one, and only one, of the enclosed values *must* be entered.
|	In command syntax, this character separates the different options available for a parameter.
CTRL-*X*	This notation indicates a two-character sequence for control characters. To enter the control character, hold down the **Control** key (often labeled CTRL) and press the character specified by *X*.

Related Documents

The Annex document set also includes the following books:

Annex Communications Server User's Guide

Describes the Command Line Interpreter (CLI) for users with terminals connected to Annex Communications Servers.

Annex II Communications Server Hardware Guide

Describes the hardware installation procedures, the ROM Monitor commands, and troubleshooting procedures for Annex II Communications Servers.

Annex IIe Communications Server Hardware Installation Guide

Describes the hardware installation procedures, the ROM Monitor commands, and troubleshooting procedures for Annex IIe Communications Servers.

Annex 3 Communications Server Hardware Installation Guide

Describes the *Annex 3* Communications Server's hardware, diagnostics, troubleshooting, and installation.

Micro Annex Communications Server Hardware Installation Guide

Describes the *Micro Annex* Communications Server's hardware, diagnostics, troubleshooting, and installation.

Annex Communications Server Release Notes

Describes the enhancements included in the current release and other issues that currently are not documented in any other manual.

Annex Communications Server Installation Notes

Describes installing the Annex operational code on a UNIX–based host for subsequent loading onto an Annex.

Chapter 1

Introduction to the Annex

General

The Annex Communications Server increases both the accessibility and the power of an Ethernet or token ring local area network. With an Annex, you can attach virtually any serial device(s) to the network. The Annex supports and manages these devices, and provides many applications for connecting users and resources on the network (see Figure A–1).

Since the Annex was specifically designed for use with UNIX systems, its user interface looks and feels like UNIX, and the Annex's networking capabilities are UNIX-compatible. These capabilities support the 4.3BSD tahoe distribution of the TCP/IP network protocols.

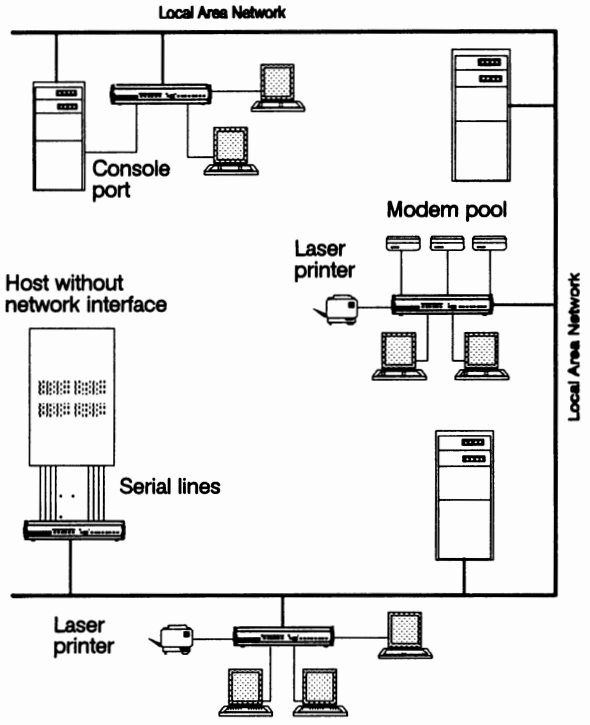

Figure A–1. Typical Local Area Network

Annex Capabilities

Devices attached to the Annex can easily access, or be accessed by, the network. The Annex software provides network management tools and routing capabilities for managing the connections between these devices.

The Annex can be transparent to a UNIX host and to the application running on that host. This transparency allows Annex-attached devices to look like directly-attached devices to the UNIX host.

This section describes the following Annex features:

- Network administrator (**na**)
- Command Line Interpreter (CLI)
- User interface
- Expedited remote procedure call daemon (**erpcd**)
- Extensive security system
- Port servers and rotaries
- UNIX host-originated connections
- Name server support
- Network management
- Routing
- Multi-protocol support

Network Administrator (na) Utility

The network administrator (**na**) program is a host-based UNIX utility. It provides commands for displaying and modifying operating characteristics of the Annex and its ports. The **na** commands allow you to boot, to produce an up-line dump before a boot, and to broadcast administrative messages to ports on an Annex.

Command Line Interpreter (CLI)

The Command Line Interpreter (CLI) is the Annex's command interface. It is what a user sees from a terminal attached to the port.

The CLI commands allow users to connect to hosts, to move back and forth between established sessions, and to display and change port characteristics. Users can also display known hosts, as well as statistics for the Annex and network. In addition, the CLI provides superuser commands for network administration and management.

The CLI **admin** command, accessed as a superuser on a CLI connection, is a local substitute for the host-resident **na** command, especially in stand-alone environments. The CLI **admin** command set provides a subset of the host-resident **na** command set.

Customizing the User Interface

Using the Annex's macros feature, you can customize the CLI user interface and set up site-specific prompts and commands, making the CLI invisible to the user. For example, you can create aliases for CLI commands that connect users directly to different hosts and/or applications. Or, create menus that hide the Annex's command interface while providing as many of the Annex's options as the user requires.

Expedited Remote Procedure Call Daemon (erpcd)

The expedited remote procedure call daemon (**erpcd**) is an Annex utility that runs on a UNIX host. It listens for requests by the Annex for a file server (download of the operational code and other files) and for a security server.

Extensive Security System

The Annex provides comprehensive security features that assist you in securing your Annexes and the network from unauthorized or undesired access. With these features, you can select between host-based security – where at least one host on the network is functioning as a security server, and local password protection – where the passwords are stored on the Annex. Optionally, you can use local password protection as a back-up to host-based security.

The Annex allows you to configure the following security checkpoints:

- CLI security: access to the Annex by a user at a device attached to a port, such as a terminal or modem.
- Port server security: access to a device attached to a port by a user at another host on the network.
- Connection security: access to hosts or networks by a user at an Annex.
- Virtual CLI security: access to a virtual CLI connection.

The Annex provides protection through the use of an administrative password that controls access to the superuser CLI commands. This password can also be used to protect access to an Annex through **na**. The security system provides audit trails to monitor users and their activities. These audit trails display user names, log-in locations, and the amount of time used. The Annex also provides the source code for the security system, and the flexibility to integrate Annex security with existing security for a network-wide system.

Port Servers and Rotaries

The port server allows the Annex to add resources to the network by allowing users and applications on the network access to devices attached to ports, through Telnet connections to the port server.

The port server supports rotaries. A rotary is a set of ports grouped together so that users can address them – and the Annex can manage them – as one resource. You can assign names to rotaries. Using rotaries, you can: assign multiple rotaries to one Annex with each rotary having its own name; assign multiple ports on an Annex to one rotary; or assign rotaries on different Annexes to one rotary name. Each rotary name can have its own Internet address and can be addressed as a separate resource on the network.

The port server also permits access to the CLI. The Annex creates a virtual CLI connection when a user at the port server requests access to the CLI. The Annex provides security capabilities for the port server, including host-based user validation or local password protection, before access to a port is permitted. The same principles of security are provided for virtual CLI connections.

The port server provides a camp-on feature. If all requested ports are busy, the port server camps on the user on a first-come, first-served basis. A user can also put the camp-on request into the background and resume another job; the Annex notifies the user when the port is free.

UNIX Host-originated Connections

The Annex provides two utilities for UNIX host-originated connections: **rtelnet** and **aprint.**

The **rtelnet** daemon is more flexible; it supports many types of applications, and establishes Telnet connections between a serial line on the Annex and a character special file on a host. This daemon is an Annex-specific reverse Telnet utility that runs on top of the pseudo-terminal facility provided by UNIX hosts; it creates host-originated connections to devices attached to the Annex serial ports. The Annex **rtelnet** utility allows a UNIX host to open, read, and write a pseudo-tty corresponding to an Annex port. Using **rtelnet**, protocols such as **tip, cu, uucp**, and **kermit** can work with modems and with PCs attached to Annexes. Also, **rtelnet** can be used with printing software (e.g., PostScript) that accesses bidirectional printers and with printing packages that expect a tty device.

The **aprint** utility has only one application: to send files directly to a printer connected to the Annex's serial port or parallel printer port. This simple print utility does not provide spooling capabilities, but can be incorporated into other script files or utilities that send UNIX files to Annex printers.

Name Server Support

The Annex supports multiple name servers on the network, including the Domain Name System (DNS) server and the IEN-116 name server. You can configure the name server you prefer as the Annex's first choice (source code for an IEN-116 name server is provided with the Annex software).

A DNS server enables a local network to connect to large IP networks, such as the Internet. The Annex uses a DNS server for:

- Multiple names for the same Internet address
- Multiple Internet addresses for the same host

The Annex can listen to, extract host names from, and build host tables with RWHO packets. Although RWHO is not a name server, the Annex can use it as one. Using RWHO is adequate for small networks in which all hosts broadcast RWHO packets.

Network Management

The Annex provides network management and host-based administration that allows you to manage hundreds of Annexes remotely from one terminal located anywhere on the network.

The CLI provides commands for performing many of the same tasks that can be performed using a line monitor or break-out box. You can issue CLI commands remotely through virtual CLI connections. CLI administrative commands allow you to:

- Tap ports
- Force control signals from low to high or high to low
- Find out who is connected and if that user is active
- Send test messages to users
- Perform remote loopback on other hosts

The Annex's host-based administration provides tools for downloading the Annex software from a file server host. In the unlikely event of software problems, you can also obtain an Annex dump while at a host. These dumps assist customer support personnel in resolving problems.

Full Routing

The Annex supports full routing that allows access to all hosts on the network, regardless of location. The network can be a small, simple network or a large campus- or organization-wide network with many subnets and gateways to other networks. The full routing on an Annex includes:

- Dynamic routing in which the Annex uses messages received from the network to learn routes.
- The ability to discard routing information when maintaining a routing database is not appropriate.
- The ability to create hardwired routes to control the routing information in smaller networks.

Multi-protocol Support

The Annex supports multiple protocols, allowing access to both TCP/IP and LAT services. The **telnet** and **rlogin** commands access TCP/IP hosts; the **connect** command accesses LAT services.

Applications for the Annex

The Annex supports many applications that go beyond simply servicing terminals. With an Annex, you can:

- Connect terminals, modems, and printers
- Connect PCs and X-windows terminals
- Connect remote hosts, network, and Annexes
- Connect hosts lacking a network interface
- Perform remote system management

Connecting Terminals

The Annex provides many options for configuring terminal behavior. The terminal can have access to the CLI, where the user can create multiple simultaneous sessions to one or more hosts. The Annex provides the ability to *hot-key* back and forth between these sessions with user-defined key sequences. Individual jobs from these multiple sessions can be put in the background. This allows messages and notifications (such as the arrival of mail) to be displayed on your terminal while you are working in another session.

Also, terminals can be configured for automatic connection to hosts, bypassing the CLI completely. These automatic connections can be set up in one of two ways:

- Dedicated port connection to one dedicated host and application
- Slave port connection, using **rtelnet,** in which the host initiates and controls the connection

Either option hides the network from less technical users. These types of configurations are used so that terminals access only one application (for example, a terminal for order entry) or terminals in areas where the public has access to them.

Connecting Remote Hosts, Networks, and Annexes

The Serial Line Internet Protocol (SLIP) allows you to connect remote hosts, networks, and Annexes to the local network. By connecting remote Annexes using SLIP, you can provide terminals and printers in remote offices with access to the main office network.

The Compressed Serial Line Internet Protocol (CSLIP) can compress the 40 bytes of TCP and IP headers to as little as three bytes when running over a SLIP link. Compression creates smaller packets, and faster throughput. When sending keystrokes over a 2400 baud modem line, it takes approximately 320 milliseconds (ms) to echo a single keystroke over a normal SLIP connection. You can choose a configuration that either uses CSLIP always, or one that uses CSLIP only when the remote end sends compressed SLIP packets.

Connecting PCs to the Network

The Annex provides two options for connecting PCs to the network:

- Running a standard terminal emulator program
- Running SLIP, including Compressed SLIP (CSLIP)

Using a terminal emulator program, a PC has the same capabilities as a terminal connected to an Annex. Additional functions, such as file transfers, may be available with the PC's software.

By running SLIP, the user can connect a PC to the network using a serial port attached to the Annex. The PC behaves as an IP host on the network, allowing host connectivity via Telnet, mail service via SMTP, and file transfers via FTP. All Internet services are available just as if the PC was connected directly to the network.

Connecting X-windows Terminals to the Network

Generally, X-windows terminals have a serial interface as well as a network interface. The serial interface can be used to connect the X-windows terminal to the network. Some X-windows vendors provide host-based software, enabling X applications to run over a serial line. Other vendors provide SLIP on the terminal for this purpose. In either case, Annex serial ports can be used to connect the X-terminals, providing full transparency to X applications.

Connecting Modems

The Annex provides many options for configuring modem behavior. A modem can be set up to make outbound calls only, to make inbound calls only, or to be bidirectional. Using **rtelnet**, an outbound modem can be accessed by **tip**, **cu**, and **uucp**.

Inbound modems can be configured with a range of restrictions – from having full access to the network through CLI, to having restricted access to a dedicated host. The behavior of bidirectional modems is defined by whether the call was initiated by the modem or by an application on the local network.

Modems attached to an Annex create a modem pool, which is easier to manage than are modems attached to several different computers. In addition, the Annex's security system adds a level of protection beyond that provided by individual hosts.

Connecting Printers

The Annex supports printers on its serial ports as well as its parallel port. The parallel port supports a printer that uses the standard Centronics interface; the Annex IIe and Annex 3 support the standard Dataproducts interface as a software-selectable option.

Using **rtelnet**, the Annex supports printers, such as PostScript, that require dynamic font downloading and bidirectional communications.

Connecting Hosts without a Network Interface

The Annex can act as a front-end to a host lacking a network interface by providing that host with the interface. By attaching the host's terminal lines to the Annex's serial ports, users on the network can access the host through the Annex using the Telnet protocol. The Annex's rotary capabilities include support for names and Internet addresses for serial ports attached to the host, and a camp-on feature when all ports are busy.

Performing Remote System Management

The Annex supports remote system management through a connection between a computer's console port and an Annex serial port. In this configuration, you can reboot and perform kernel debugging remotely on systems to which you do not have physical access.

Using this Manual

This manual is organized into four books:

- *Book A: Overview*
- *Book B: Configuration Procedures*
- *Book C: Network Management*
- *Book D: Reference*

Book A: Overview presents a general introduction to the Annex, its features, and its applications. It contains two chapters: *1: Introduction to the Annex*, and *2: Network Protocols*.

Book B: Configuration Procedures describes how to configure the Annex and its ports. It contains seven chapters: *1: Configuring Annexes*, *2: Configuring Ports*, *3: The Port Server and Rotaries*, *4: Printers*, *5: Modems*, *6: Serial Line Internet Protocol (SLIP)*, and *7: Configuring Hosts and Servers*.

Book C: Network Management describes using the Annex's utilities to manage Annexes and the network. It contains two chapters: *1: Network Administration*, and *2: Simple Network Management Protocol (SNMP)*.

Book D: Reference provides a detailed reference for all tools, utilities, and daemons supplied with the Annex software. This book contains four chapters: *1: na Commands*, *2: na Parameters*, *3: CLI Commands*, and *4: Utilities*.

Because this manual is organized around tasks, information may be located in more than one place. For example, to configure security for an Annex, use *Book B: Configuration Procedures* and refer to:

- *Using Annex Security* in Chapter 1, which describes the setting the parameters Annex security.
- *Port Security* in Chapter 2, which describes the available port security mechanisms, and how to set port parameters.
- *Port Server Security* in Chapter 3, which describes security for the port server and for a virtual CLI connection, and how to set port parameters.
- *Host-based Security* in Chapter 7, which describes how to set up the security server(s) and the required host files.

Chapter 2

Network Protocols

General

Network management is a major aspect of Annex administration. This chapter describes the Annex's implementation of local area network (LAN), TCP/IP, and LAT protocols. These protocols are international standards; however, networks can choose to implement them differently.

Local Area Network Protocols

The Annex supports two local area network protocols: Ethernet Rev. 2.0/IEEE 802.3 and IEEE 802.5/Token Ring. These protocols provide both the data link and physical layers for communication between the Annex and the local host.

Ethernet/IEEE 802.3 Protocol

The Ethernet and IEEE 802.3 protocols specify the method for accessing the physical and network levels of the transmission medium. At the physical level, access is provided via Carrier Sense Multiple Access with Collision Detection (CSMA/CD). At the network level, Ethernet and IEEE 802.3 specifications handle IP encapsulation differently. Although the differences are minor, these methods are not compatible; hosts using one encapsulation method cannot communicate with hosts using the other. The Annex can be configured to use either method; it is compatible with both Ethernet Rev. 2 and IEEE 802.3 transceivers.

The transmission medium is a common bus with high transmission data rates (10 MHz). It is often a coaxial cable; however, other media, such as fiber optic cable, twisted pair, or thin-wire Ethernet can be used. Because this medium is a common bus, Ethernet permits multiple hosts to share the same transmission medium.

When one station wants to transmit data, it listens for traffic on the medium. If there is none, the station transmits the data. If there is traffic, the station waits until the transmission has completed. If two stations transmit simultaneously, both stop transmitting and then each wait a different length of time to retransmit.

IEEE 802.5/Token Ring Protocol

The IEEE 802.5 protocol also specifies how to access a physical medium. This access method is known as the Token Ring Access Method. The Annex supports both IEEE 802.5 and the IBM Token Ring architecture.

A token ring is a series of stations connected serially (point-to-point) over a transmission medium. This medium can be a coaxial cable, a twisted pair, or a fiber optic cable. Information is transferred sequentially from one station to the next.

A station must have possession of the *token* in order to transmit. The token is a control signal that circulates on the medium between information transfers. Any station, upon detecting the ring, can capture the token for the purpose of transferring information. When information transfer is complete, the station generates a new token.

Hardware Addresses

Each network interface has a unique hardware address. To maintain uniqueness, hardware addresses are managed by the IEEE, which assigns blocks of addresses to manufacturers of LAN devices. Each manufacturer assigns a unique address to each device built. A hardware address consists of six octets of hexadecimal digits, for example: 00-08-2D-00-00-37.

The Annex's hardware address is assigned at the factory; it is permanently stored in ROM. You can display the Annex's hardware address using either the ROM Monitor or the CLI **stats** command. Sometimes, hardware addresses are used for testing the local area network.

TCP/IP Protocols

TCP/IP is a set of protocols that are part of the Internet Protocol suite. These protocols define the network and transport layer. The Annex also implements some protocols at the application layer. The Internet Protocol suite evolved from work sponsored by the Defense Advanced Research Projects Agency (DARPA) for the development of a wide area network to interconnect multiple networks and hosts. The end result of this work became the DARPA Internet, a wide area network that ranges from networks with a small number of hosts to networks with a large number of hosts to networks interconnected with other networks. These interconnections constitute an Internet.

The Annex supports the latest enhancements provided in the 4.3BSD tahoe distribution of UNIX. These include enhancements, such as subnet mask and broadcast addresses, required by changes in the Internet protocols after the release of 4.2BSD.

Internet Protocol Addressing

An Annex is a host on the Internet. For the network layer (IP) to route packets to the Annex, it requires a unique Internet address. Typically, an Internet address is a 32-bit address divided into four 8-bit fields, with each field separated by periods, and specified as a decimal number (from 0 to 255), a hexadecimal number, or a combination of both. For example, all entries in the next example specify the same address; this format is called dot notation:

192.9.200.100
0xC0.0x9.0xC8.0x64
192.9.200.0x64

Note: The Internet address is displayed in decimal dot notation with commands.

Network Address to Name Translation

The Annex supports the use of host names, which permits users to enter a symbolic name rather than the Internet address when requesting a connection to, or information about, a specific host. The Annex can translate a host name to an Internet address. The Annex provides this translation using a *host table*. The host table maintains host names and Internet addresses for known hosts on the network. These names and addresses are provided by name servers. A name server is a host on the network that supplies Internet addresses for host names and names for addresses.

The Annex supports two widely-used name servers: IEN-116 and Domain Name System (DNS). You can implement one or both of these name servers for the Annex.

The Annex also monitors RWHO broadcasts. The Annex stores information in the host table from these broadcasts. RWHO is a BSD protocol for passing host information around the network.

Network Address to Hardware Address Translation

To transmit an IP packet on the local area network, the Annex maps an Internet address to a hardware address, using the Address Resolution Protocol (ARP). This protocol dynamically maps between these addresses. As mappings are acquired, they are stored in an ARP table. The information in this table can be displayed and modified.

When an Annex has data to send to an Internet address, and it does not have a hardware address in its ARP table for that Internet address, it broadcasts an ARP request for a hardware address for the Internet address associated with the data to be sent. If a response to the ARP request is received, the new mapping is stored in the ARP table and the data is sent to the appropriate destination.

Network Classes

The Internet protocol divides the address into a network section and a local or host section. The address has five classes, of which three can be used for hosts such as Annex. The class for the address is determined by the number of bits assigned for the network section of the address.

- The first address type is Class A and has a 7-bit network number and a 24-bit host address. This allows 128 Class A networks.
- The second address type is a Class B and has a 14-bit network number and 16-bit host address. This allows 16,384 Class B networks.
- The third address type is a Class C and has a 21-bit network address and a 8-bit local address. This allows 2,097,152 Class C networks.
- The fourth and fifth address types are Class D and Class E. These cannot be used for normal host applications.

Table A–1 lists the three network classes, the decimal number that appears in the first octet, and what sections of the Internet address are assigned to the network and to the host. The *nnn* represents all or part of the network number and the *hhh* represents all or part of the host address.

Table A-1. Network Classes

Class	First Octet	Internet Address
A	1-126	nnn.hhh.hhh.hhh
B	128-191	nnn.nnn.hhh.hhh
C	192-223	nnn.nnn.nnn.hhh

Note: The Class A network 127 is reserved for software loopback; do not use it as an Internet address.

For example, the Internet address 63.000.000.035 is a Class A address; *63* is the network address and *35* is the host address. With a Class A network, the Internet address can be specified as 63.0.0.35 or 63.35.

The Internet address 129.091.000.063 is a Class B address. It can be specified as 129.91.0.63. The network address is represented by the decimal numbers *129.91*, and the host address is *63*.

The combination of network address and host address is used to maintain a unique Internet address for each host. For networks interconnected to other networks, such as the Internet, both the network address and the host address must be unique. In this case, assignment of both the network and the host address is usually administered centrally.

For a private network, host addresses can be selected arbitrarily, but they still need to be centrally administered. Network addresses, even for a private network, should be officially assigned. Internet addresses can be obtained from the Network Information Center (NIC).

Subnet Addressing

Another provision of Internet addressing is the subnet. A subnet allows several interconnected local networks to share a single network number. The local network could include multiple physical networks (such as Ethernet, ring nets, or point-to-point links) but appears to the external network as a single entity.

Addressing for a subnet is implemented with the host address section of the Internet address. That is, part of the host address is given a subnet number. Within a subnetted network, each subnet appears as a distinct network. In order for a given host within a subnet to determine whether a packet is to be routed in or out of the subnet, a subnet mask is used.

The subnet mask defines for a host what part of the entire Internet address is to be used for the network address, the subnet address, and the host address. By convention the subnet address is the part of the host address adjacent to the network address.

For example, a local network that uses the network address of 129.91 has created two subnets. By agreement within this local network, the subnet is specified as the second octet. That is, all hosts assigned to one subnet use the Internet address of 129.91.1.*x*, where *x* is the host address; hosts assigned to the other subnet use 129.91.2.*x*. In this local network, each host has a subnet mask defined as 255.255.255.0. This mask indicates that the first and second octets are the network addresses, the third octet is the subnet address, and the fourth octet is the host address.

For subnet addressing to work in any local network, all hosts on the network must have the same subnet mask defined. Also, each host must have the same network address. Hosts can be informed of the subnet mask in one of three ways: 1) defined in the configuration, 2) obtained from the host's Internet address, and 3) sending out an ICMP Address Mask Request and receiving a reply from an authoritative agent.

The Annex can be configured as an authoritative agent on the network and reply to ICMP Address Mask Requests.

Broadcast Addresses

The Annex provides the ability to configure the broadcast address. The standard broadcast address uses a host address of all ones; that is, *network*.255 or *subnet*.255. This style is used with 4.3BSD hosts. Alternatively, 4.2BSD networks use a broadcast address of zeros (or *network*.0) for the host address.

The Annex's broadcast address can be configured for either type of network requirement. If the local network does not support a 4.3BSD-style broadcast address, the broadcast address for an Annex can be defined as zeroes (*network*.0). This supports both 4.2BSD and 4.3BSD broadcast addressing.

Routing Services

In a simple network of a single LAN cable in which all hosts are attached, routing is not a requirement. However, a simple local network can grow to multiple networks connected by routers or gateways. At this point, routing services become a necessity to reach hosts located on distant networks. Routing services inform hosts on one network of the route to a host on the other network.

The Annex can use static (or fixed) route entries and can listen for dynamic routing updates. The dynamic routing updates involve the Annex using a routing daemon similar to the BSD routing daemon (**routed**) for its routing services. This daemon uses Routing Information Protocol (RIP) messages to create routing tables. RIP messages are sent by a gateway announcing that it provides the route to a destination. The routing tables created from these RIP messages include entries for Internet gateways to hosts on other networks and to those other networks.

In addition to RIP messages, the Annex can receive ICMP *redirects*. Redirects are messages used by ICMP to announce that the expected route to a destination is not correct and to supply the correct route. The Annex uses the redirects to update its routing table. Although the Annex can create entries in its routing table by listening for RIP broadcasts, it cannot broadcast RIP messages.

Internet Trailer Packets

An Annex on an Ethernet will support the trailer packets used in 4.2BSD UNIX. Trailer packets are not recommended, as they are more overhead for the Annex. A host will use trailers for 512-byte blocks only. An Annex running on an IEEE 802.5/Token Ring network does not support trailer packets.

Local Area Transport (LAT) Protocol

The Local Area Transport (LAT) protocol multiplexes low-speed user data over a high-speed local area network (LAN). Because LAT is an Ethernet-based communications server protocol, its architecture bypasses the LAN's session, transport, and routing layers. This frees up CPU cycles by off-loading much of the communications services management from the host.

LAT Architecture

LAT architecture consists of two layers: the virtual circuit layer and the slot layer.

Virtual Circuit Layer

The virtual circuit layer provides a link to all hosts communicating with the Annex; it also provides a data transport service for the slot layer. One message exchange establishes a virtual circuit. The virtual circuit layer sends a packet at least every 30 milliseconds (ms). The default value for this parameter is 80 ms.

If a virtual circuit packet is not acknowledged within the time frame specified by the circuit timer parameter, the Annex waits approximately one second. The Annex, after a predetermined number of retries fail, assumes that the host is down.

If there is no data to send, the virtual circuit goes into a "balanced" mode: either side can re-initiate transmission if it has data to send. By default, a *keepalive* message is sent every 20 seconds.

Slot Layer

The slot layer is built on top of the virtual circuit layer, and has three functions: 1) establishing user sessions, 2) multiplexing sessions over a virtual circuit layer, and 3) providing a transparent data transfer access service to LAT users.

LAT uses five slots: **Start** and **stop** slots control sessions; the **attention** slot is used for out-of-band signalling; the **A** slot is used for data transfer; and the **B** slot is used for transmitting physical port and session characteristics.

A single packet at the virtual circuit layer can contain several slots. The LAT protocol multiplexes data between the Annex and a particular host by combining several characters per user in a slot, and several users in a virtual circuit packet.

Service Advertisement

Every host that accepts communications sessions is a service provider. All service providers broadcast the availability of their service, along with a current service rating, approximately every 60 seconds.

A node can provide multiple services. When a user broadcasts a service request and there are multiple providers of that service, the Annex logs the user onto the node with the highest service rating. Typically, four factors account for a given service rating: 1) the most recent CPU idle time, 2) the CPU type, 3) the amount of memory, and 4) the number of available interactive slots.

Group Codes

Service providers can be assigned a LAT group code. Group codes partition the LAT network logically into subsets. The Annex restricts clients to the assigned group code(s). Group codes can enhance both network security and network management.

Chapter 1

Configuring the Annex

General

Configuring the Annex involves setting **na** parameters to define the unit's necessary operating and administrative attributes. The types of administrative attributes you can configure include:

- Defining Internet addresses for the Annex
- Defining the preferred hosts for booting and dumping
- Setting up security for the Annex
- Setting up the use of name servers
- Setting up the use of event logging
- Setting the local time zone for using a time server
- Customizing the Annex
- Setting the token priority for a token ring network
- Configuring LAT

See *Book D: Reference, Chapter 1: na Commands* and *Chapter 7: Configuring Hosts and Servers* in this book for more information.

Configuring an Annex Using the na Utility

The **show annex** command displays the current settings for Annex parameters. The **set annex** command allows you to change any setting. All parameters have default settings. Some of these parameters must be set using the ROM Monitor before booting the Annex with its operational code.

Note: By default, the **show annex** command scrolls the selected parameters line by line in two-column format. To use a *pager* along with the **na** command **show**, set the following variables in your host environment:

- PAGER = *<name your choice of pager>* (e.g., more, less, pg)
- XY_NAPAGER = Y

To configure an Annex:

1. Execute the **na** program as follows:

```
% na
Annex network administrator R6.0 November 3, 1990
command:
```

2. Specify the Annex. For example:

```
command: annex 132.245.6.40
```

   Or specify multiple Annexes. For example:

```
command: annex 132.245.6.40, frontlobby
command: annex
enter default annex list:132.245.6.40, frontlobby
```

3. Execute the **set annex** command for those parameters you want to change. The following example selects these options:
   - Enable the use of DNS as the name server
   - Define two name server hosts
   - Enable security on the Annex
   - Define the security server host
   - Enable security for virtual CLI connections
   - Define an administrative password
   - Enable event logging
   - Define a CLI prompt

```
command: set annex name_server 1 dns\
         pref_name1_addr 132.245.6.95\
         pref_name2_addr 132.245.6.85\
         enable_security Y\
         pref_secure_host 132.245.6.95\
         vcli_security Y password piano\
         syslog_mask all\
         syslog_host 132.245.6.95\
         cli_prompt "%n%s%p%c"
```

4. Execute the **show annex** command to review your changes. Using the above example, the terminal would display:

```
command: show annex=frontlobby

       annex frontlobby:
              inet_addr: 132.245.6.38
        pref_load_addr: 132.245.6.95
        pref_dump_addr: 132.245.6.95
        load_broadcast: Y
            image_name: "oper.16.enet"
           subnet_mask: 255.255.255.0
 authoritative_agent: Y
      broadcast_addr: 0.0.0.0
   load_dump_gateway: 0.0.0.0
       name_server_1: dns
     pref_name1_addr: 132.245.6.95
       name_server_2: none
     pref_name2_addr: 132.245.6.85
nameserver_broadcast: N
     host_table_size: 64
   pref_secure1_host: 132.245.6.95
   pref_secure2_host: 132.245.6.80
  security_broadcast: N
       vcli_security: Y
  network_turnaround: 2
     enable_security: Y
  load_dump_sequence: net
        ipencap_type: ethernet
   server_capability: none
         syslog_mask: all
    syslog_facility: log_local7
        syslog_host: 132.245.6.95
         cli_prompt: "%n%s%p%c"
          motd_file: "motd"
timezone_minuteswest: 300
    daylight_savings: usa
      time_broadcast: N
            max_vcli: unlimited
            password: "<set>"
             acp_key: "<unset>"
       vcli_password: "<unset>"
              routed: Y
               rwhod: Y
min_unique_hostnames: Y
       ring_priority: 0
        sys_location: ""
             lat_key: ""
```

5. Execute either the **boot** or **reset annex all** command to effect these changes at the Annex.

You can configure more than one Annex simultaneously using one of these options:

- Define all Annexes you want to configure using the **annex** command, then use the **set annex** command to change the parameters.
- Define the parameters for one Annex and use the **copy annex** command to copy the Annex parameters to the other Annexes.
- Define the parameters for one Annex and use the **write** command to create a script file with all configuration data for that Annex. Next, execute the **read** command for all Annexes you want to configure.

Note: The **write** and the **copy annex** commands do not write or copy the Annex's Internet address, administrative password, virtual CLI password, LAT key, or access control protocol key.

Annex Internet Addressing

The Annex uses Internet addressing to communicate with hosts on the network. Internet support requires an Internet address, a broadcast address, and a subnet mask.

The Internet Address

The Annex's Internet address is a 32-bit address divided into four 8-bit fields. Each field is separated by periods and is specified as a decimal number (from 0 to 255), a hexadecimal number, or a combination of both. In the following example, all entries specify the same address. This format is called *dot notation*:

192.9.200.100
0xC0.0x9.0xC8.0x64
192.9.200.0x64

Note: The Internet address is always displayed in decimal dot notation with the CLI, **na**, or ROM Monitor commands.

The Annex's Internet address is defined with the **inet_addr** parameter. This address must be set prior to down loading the operational code to the Annex. Therefore, it is always specified with the ROM monitor **addr** command during the Annex's initial installation.

The Broadcast Address

The broadcast address defines the Internet address the Annex uses to broadcast. The Annex will broadcast requests when it has not received a response from a server, such as file server or security server. The **broadcast_addr** parameter specifies this address.

Set the broadcast address according to the network's broadcast schemes. Earlier broadcast schemes specified that the host part of the address is assigned all zeros (or *network*.0). The current broadcast scheme specifies that the host part of the address is assigned all ones (or *network*.255).

The Subnet Mask

If the network is divided into subnets, you must specify the Annex's Internet subnet mask with the **subnet_mask** parameter. You define the network address and the subnet address with the decimal value of 255 in the subnet mask. For example, if the network address is the first octet and the subnet address is the second octet, the subnet mask would be specified as 255.255.0.0.

If you do not define the subnet mask, the Annex assigns one based on the network part of its Internet address. This parameter is normally set with the ROM Monitor **addr** command during the Annex's initial installation.

By default, the Annex acts as an authoritative agent in respect to ICMP Address Mask Requests. If another host broadcasts this message querying for the subnet mask, the Annex replies with the subnet mask. Optionally, you can prevent the Annex from responding by setting the **authoritative_agent** parameter to **N**.

Booting and Dumping

The Annex always obtains operational code by down loading it over the network from a UNIX host that runs Annex file server software or from another Annex configured as a boot server (running the same operational code). The Annex boots each time it is powered up and upon receipt of a **boot** command.

The Annex can dump to a UNIX file server. The Annex performs a dump on demand upon receipt of either the **na dumpboot** or the superuser CLI **boot -d** command, or automatically when it detects fatal internal errors or failures.

Setting the Preferred Load Host

The **pref_load_addr** parameter specifies the preferred load (or file server) host. This is the host from which the Annex first requests a down-line load of its operational code. If this parameter is not defined or the specified host is not available, the Annex broadcasts its boot request and loads operational code from the first host that responds.

You may have defined this host before you initially loaded the Annex software using the ROM Monitor **addr** command. You can modify the preferred load host with the **pref_load_addr** parameter. Specify the host by its Internet address or its name.

The **image_name** parameter specifies the name of the image file that contains the Annex's operational code. The Annex assumes this file to be in the directory **/usr/spool/erpcd/bfs** on the file server host.

If the load host has a different network or subnet address, you must define a gateway through which the Annex can reach the host. The **load_dump_gateway** parameter specifies the Internet address for the gateway. (Original Annexes cannot load or dump across a gateway.)

During the initial boot of the operational code, the ROM Monitor requires the address of a gateway if the specified load host is on another network or has a different subnet address. In this case you enter the gateway's address with the ROM Monitor **addr** command. The Annex automatically adds this gateway to its routing table (see *Chapter 7: Configuring Hosts and Servers* in this book for more information on the routing tables).

Setting the Preferred Dump Host

The **pref_dump_addr** allows you to specify the preferred host to which the Annex performs a dump. If this parameter is not defined or the specified host is not available, the Annex broadcasts its dump request and dumps to the first host that responds.

The dump creates a file, between one and three megabytes in size, in a directory named **/usr/spool/erpcd/bfs** and assigns a unique dump file name to each Annex. In this book, *Dump Services* in *Chapter 7: Configuring Hosts and Servers* describes how file names are assigned. If the dump host has a different network or subnet address, you must define a gateway through which the Annex can reach the host. The **load_dump_gateway** parameter specifies the Internet address for the gateway.

Setting an Annex as a Load Server

The **server_capability** parameter defines the Annex as a file server host. An Annex can provide operational code only for another Annex that shares the same image of the operational code. When an Annex boots, it uses the image file to load the operational code, uses the gateways file to initialize the routing table, and uses the rotaries file to initialize the rotaries. The Annex does not normally store these files because they use memory. A file server host Annex uses approximately 60 Kbytes for the operational code; for the **gateways**, **rotaries**, **macros**, and message-of-the-day files, it uses the amount of disk space used on the host from which these files are read.

The **server_capability** parameter requires you to define the files that the server Annex supplies during a boot. Allowable values are:

all	Copies of the operational code, the **gateways** file, the **rotaries** file, the message-of-the-day file, and the **macros** file.
gateways	A copy of the **gateways** file.
image	A copy of its operational code.
macros	A copy of the **macros** file.
motd	A copy of the message-of-the-day file.
none	The Annex is not a file server. This is the default.
rotaries	A copy of the **rotaries** file.

Note: If you configure an Annex to supply only a copy of the operational code, the default is for the Annexes being booted to broadcast for the **gateways**, **rotaries**, message-of-the-day, and **macros** files.

Disable Broadcasting for Files during a Boot

During a load, the Annex broadcasts for the **gateways, rotaries, macros,** and message-of-the-day files if they are not available on the preferred load host. You can disable broadcasting for these files by setting the **load_broadcast** parameter to **N.**

Using SLIP for Booting and Dumping

You can load and dump an Annex over the local area network (Ethernet or token ring) or over a serial line using the Serial Line Internet Protocol (SLIP). The default is to use the local area network. The **load_dump_sequence** parameter specifies which network interfaces is to be used for a load or a dump and the order they are to be used.

Define the local area network with the value **net.** To define a SLIP line, use the value **sl***nn*, where *nn* is the number of the serial port. You can enter up to four interfaces with this parameter. Each interface must be separated by a comma. For example:

```
command: set annex load_dump_sequence sl2,sl6,net
```

Note: Any serial line you define with this parameter **must** be configured for SLIP (see *Chapter 6: Serial Line Internet Protocol (SLIP)* in this book).

The Annex supports booting or dumping across a SLIP interface in non-compressed SLIP only. Original Annexes do not support this feature.

TFTP Loading Protocol (Operational Code)

TFTP is an alternate loading protocol for ERPC. The Annex operational code opens and reads the operational image (for AOCK strings), **gateways, rotaries, macros,** and message-of-the-day files. The Annex accesses each file one at a time in turn.

The Annex initially tries to open a file via ERPC. If ERPC fails or times out, the Annex tries to open a file via TFTP. If the TFTP request fails or times out, the Annex retries opening the file via ERPC. This cycle continues until the Annex succeeds in opening the file or until the Annex reaches a maximum try count (currently 8 cycles). If the **load_broadcast** Annex parameter is enabled and the Annex cannot open a file from the **pref_load_host,** the Annex broadcasts the open request (this is true for both the ERPC and TFTP protocols). Once a file is successfully opened, the Annex continues to read the file via the protocol with which it was opened.

The protocol used to transfer any file is independent of the protocol used to transfer any other file. For environments that support both ERPC and TFTP, this means that the Annex may use TFTP to transfer one file and use ERPC to transfer another file.

Using Annex Security

The Annex provides a security system that allows you to implement as many security measures as the network requires. You can set up the security subsystem to use host-based, local password protection, or a combination of the two. In addition to these security mechanisms, the Annex provides an administrative password that is used to validate access through the administrative tools.

Note: If your Annex is accessible in any way to unauthorized users, we strongly suggest that you enable the security features after loading the host code and booting the unit.

Host-based security uses the Annex's Access Control Protocol (ACP) and requires that a host function as a security server. You can modify the host-based security policy supplied with the Annex to implement a security policy that fits the needs of your environment. Host-based security allows you to define:

- User validation on access to and from the Annex.
- Connection protection to other hosts and networks.
- User activity logs.
- Encryption of security messages between the Annex and the server.

Local password protection can be defined for access to and from individual ports and for virtual CLI access. Local password protection does not provide logging of security events to the security server. If event logging is enabled, user activities can be logged to **syslog** with local password protection. See *Chapter 2: Configuring Ports* and *Chapter 7: Configuring Hosts and Servers* in this book for more information.

Configuring the Annex for security involves:

- Enabling security for host-based security, local password protection, or both.
- Defining the server if you are using host-based security.
- Defining a local virtual CLI password for local password protection.
- Defining the administrative password.

Enabling Security

To use any security feature, you *must* enable security for the Annex by setting the **enable_security** parameter to **Y**. This parameter is mandatory if you intend to use any Annex security mechanisms except the administrative password for access to administrative tools.

Setting Up Host-based Security

Host-based security requires that at least one host on the network acts as a security server. The security server maintains files that contain user validation and connection permission information. Also, the security server can encrypt security messages.

Specifying the Security Hosts

The **pref_secure1_host** and **pref_secure2_host** parameters specify the preferred security hosts. The Annex first queries the **pref_secure1_host** for user validation or permission connection requests. If a response is not received within a limited amount of time, the Annex repeats the query several times. If the Annex still does not receive a response, it queries the host defined with the **pref_secure2_host** parameter. If a response is not received from the second security host within a limited time, the Annex broadcasts. If the broadcast fails, the Annex denies the user's request.

Restricting telnet Access to Certain Ports

The Annex operational code passes the TCP port number to the ACP when doing a **telnet** on a port with **connect_security** enabled. This feature allows restrictions on connections to certain TCP ports. The port number is available in **annex_to_port** () in the **acp_policy.c** file.

The **network_turnaround** parameter specifies the amount of time in seconds in which the Annex expects a response from the security servers. In order to reduce the possibility of a retry, the network turnaround time should be sufficient to allow for a network transmission to the security server and transmission back to the Annex.

Disable Broadcasting for Security Servers

The Annex automatically broadcasts for a security server when the host defined using either the **pref_secure1_host** or **pref_secure2_host** parameter does not respond. You can disable broadcasting for a security server by setting the **security_broadcast** parameter to N.

Virtual CLI Protection

You can set up security for virtual CLI connections in which users must provide a valid user name and password before they are granted access to a virtual CLI. You can define security on the Annex's virtual CLI connections using the **vcli_security** parameter.

Note: Setting **vcli_security** to **Y** has the same effect as enabling **cli_security** and **connect_security** on a serial port.

To set up virtual CLI protection with host-based security:

- Enable security: set the **enable_security** parameter to **Y**.
- Enable virtual CLI security: set the **vcli_security** parameter to **Y**.
- Create a password file on the security server (see *Chapter 7: Configuring Hosts and Servers* in this book).

Note: The original Annex does not support the **vcli_security** password. Virtual CLI security on an original Annex is enabled automatically when the **enable_security** parameter is set to **Y**.

Encrypting Security Messages

With host-based security, passwords are continuously sent across the network. In order to prevent unwanted access to these passwords, you can request encryption of messages between the Annex and the security server.

The **acp_key** parameter specifies the encryption key the Annex uses to exchange messages with the security server. The security server maintains the encryption key for each Annex in the **acp_keys** file (see *Chapter 7: Configuring Hosts and Servers* in this book). This encryption key also validates the security host. That is, the host must know the Annex's ACP key for the Annex to consider the host valid. Without the appropriate key, the Annex denies the user's request.

Note: The **show annex** command does not display the value of the **acp_key** parameter. Instead, it displays "**< set >**" or "**< unset >**".

Local Virtual CLI Password

Local password security allows you to assign passwords that a user must enter before accessing a port or an Annex. Since these passwords are stored locally on the Annex, they do not require a remote security server. Local password security can be used as a back-up security mechanism in case the host-based security servers are unavailable.

Local password security can be implemented for the Annex in one of two ways: upon virtual CLI connection and upon access through administrative utilities. The **vcli_password** parameter allows you to define a local password for virtual CLI connections. With this security mechanism, the user enters only the password stored on the Annex, not a user name and password. To configure the Annex for a local virtual CLI password:

- Enable security: set the **enable_security** parameter to **Y**.
- Disable virtual CLI security: set the **vcli_security** parameter to **N**.
- Define a password using the **vcli_password** parameter.

Note: Original Annexes do not support local password protection.

You can also use the **vcli_password** as a back-up to host-based security. When this local virtual CLI password is used as a back-up, the Annex first accesses the security server to validate a CLI connection request. If no response is received from a security server, the Annex requests the local CLI password. The user can enter either the virtual CLI password or the Annex administrative password.

To set up the local virtual CLI password for back-up security:

- Enable security: set the **enable_security** parameter to **Y**.
- Enable virtual CLI security: set the **vcli_security** parameter to **Y**.
- Define a password with the **vcli_password** parameter.
- Create a password file on the security server (see *Chapter 7: Configuring Hosts and Servers* in this book).

Note: The **show annex** command does not display the value of the **vcli_password** parameter. Instead, it displays "**< set >**" or "**< unset >**".

Annex Administrative Password

The Annex administrative password always validates access to superuser CLI commands; it can also validate **na** access to Annexes if security is enabled (**enable_security** is set to **Y**).

The **password** parameter modifies the Annex's administrative password. The default password is the Annex's Internet address.

Note: The administrative password is never displayed. If you forget the modified password, you can reset it only by erasing all of the Annex's nonvolatile memory using the ROM Monitor.

The administrative password always is required to access the superuser CLI commands. That is, when you enter the **su** command at the CLI prompt, you are asked for a password. If you have not modified the administrative password, enter the Annex's Internet address at the prompt.

The administrative password validates access to an Annex through **na** only when security is enabled and the password is defined. When issuing the **annex** command or specifying an Annex using **na**, you must enter this password. The password also encrypts messages between the Annex and **na**.

The administrative password can be used as the virtual CLI password and to override the password assigned using the CLI **lock** command.

The **show annex** command does not display the value of the **password** parameter. Instead, it displays "**<set>**" or "**<unset>**". When the **password** parameter is displayed as "**<unset>**", the Annex's Internet address is the default administrative password and must be used to access the superuser CLI commands.

Using Name Servers

Name servers allow users to enter names in place of addresses in order to access a host or other entity on the network. The Annex supports two standard types of name servers: a Domain Name System (DNS) server and IEN-116 server. In addition, the Annex can use RWHO broadcast messages to provide name-to-Internet address translation. You can configure the Annex to use one of these, a combination, or none.

The Annex supports the minimum uniqueness feature when entering host names. This feature allows users to enter the host name with a minimal string that is unique enough to identify that host from any other in the host table. If this feature is not enabled, the user must enter the complete name to access a host.

Setting up an Annex for name servers involves using **na** to:

- Specify the type of name server to be used and on which host(s)
- Enable or disable the Annex's RWHO process
- Manage the host table
- Enable or disable minimum unique host names

Defining Name Servers

The Annex supports two standard name server protocols: Domain Name System (DNS) and IEN-116 server. Both of these name server protocols are available in the UNIX environment. You can use one or both on the network, and the Annex allows you to specify the preferred protocol. If you choose not to use either protocol, you can configure the Annex to build the host table by listening to RWHO broadcasts.

Domain Name System

Domain Name System (DNS) servers use a distributed database to maintain host names and Internet addresses for network hosts. DNS provides a full range of capabilities that enable its use in very large networks, such as the Internet.

Each DNS server is responsible for maintaining information on all hosts in its domain. If the server receives a request for a host that is not in its domain, the server retrieves the information from another domain server for the requesting host.

A number of DNS servers are available and the Annex can support them all. One typical DNS server is the Berkeley Internet Name Domain (BIND) server. The BIND server is a standard part of 4.3BSD. Descriptions of it are available in 4.3BSD documentation. DNS provides:

- Address to name translation
- Multiple aliases for a host
- Multiple addresses for the same host

Address to name translation allows a host to obtain a name for a specific Internet address, allowing an Annex to learn its name from a DNS server. The DNS's capabilities for assigning multiple aliases or multiple Internet addresses to a single host allow you to assign multiple names to a rotary or multiple Annexes to the same rotary (see *Chapter 3: The Port Server and Rotaries* in this book for more information).

IEN-116 Name Server

The IEN-116 name server is a simple host-resident name server that uses the local **/etc/hosts** file as a database. One host is designated as the name server host, and other hosts query that host for an address. Using this method, every host on the network does not need its own up-to-date **/etc/hosts** file, and every host does not have to run **rwhod**.

The Annex distribution medium supplies the source for IEN-116 (see *Chapter 7: Configuring Hosts and Servers* in this book for installation instructions).

Setting na Parameters

The **name_server_1** and the **name_server_2** parameters define the preferred name server and the order in which to query the name servers. The name server specified with **name_server_1** parameter is the primary name server and is queried first. If the name server host is not available, the Annex queries the name server specified with **name_server_2**. You assign these parameters a value of either **dns, ien_116,** or **none,** the default.

The **pref_name1_addr** and the **pref_name2_addr** parameters define the host's Internet address that is providing the name server. The **pref_name1_addr** parameter is the host's Internet address specified in **name_server_1.**

The **pref_name2_addr** parameter defines the Internet address of the host where the name server specified with the **name_server_2** parameter resides. If **name_server_2** is set to **none,** the address specifies the second choice for **name_server_1.** This host is queried if the preferred host at **pref_name1_addr** does not respond.

Broadcasting for a Name Server

The Annex by default does not broadcast for a name server if the preferred name servers do not respond. However, you can configure the Annex to broadcast requests for a name server by setting the **nameserver_broadcast** parameter to **Y.** You may wish to use broadcast as a back-up for a name server.

Leaving name server broadcasting disabled reduces network traffic, as many networks have multiple name servers. Broadcasting for a name server when your local network is connected to a wide-area Internet can cause a tremendous amount of traffic on your local network.

Using the RWHO Protocol

RWHO is a BSD protocol that hosts use to pass information about themselves to other hosts. This information includes the host's name, who is logged in, up time, and load factor. The RWHO daemon, **rwhod,** broadcasts this information and listens for RWHO messages from other hosts, storing what it receives in a file. The information can be displayed with the **rwho** and **ruptime** commands from a UNIX host.

The Annex uses the RWHO protocol as a name server. The Annex runs an **rwhod** that listens for broadcasts from other hosts, but does not broadcast information about itself. When the Annex receives an RWHO message, it stores the host name, status information, and the source address from the IP header as the host's Internet address in its host table.

Using only RWHO messages to build the host table is often satisfactory for small networks where all the hosts run **rwhod**. Since **rwhod** imposes a load on hosts, it is frequently not used in networks primarily comprised of workstations. In large or heavily loaded networks, RWHO broadcasts can impose an excessive load on the network.

Some hosts send RWHO packets with incomplete source addresses in the IP header. The Annex is unable to store an Internet address for these hosts; causing the host table to display the host's Internet address as "-.-.-.-".

If an **rwhod** forwards packets from one network to another, the Internet address in the IP header is that of the forwarding host, not of the host whose name is in the data packet. This results in the Annex storing the wrong Internet address for that host.

Because the Annex does not broadcast RWHO messages, Annex names never appear in host tables built exclusively from these broadcasts. In which case, the only way to access an Annex using the **telnet** command is with an Internet address.

The **rwho** parameter defines whether or not the Annex listens for RWHO broadcasts. Setting the parameter to **N** disables the Annex's **rwhod** and prevents the Annex from using RWHO messages for building the host table. The default is **Y**.

Managing the Size of the Host Table

When the host table acquires a new entry after it is full, the Annex deletes the oldest, least-used entry to make room for the new one. If the host table size is too small, the host table is inconsistent due to frequent changes. You can increase the size of the host table to reduce this problem.

You modify the host table size with the **host_table_size** parameter. This parameter specifies the number of entries in the host table. You can specify the size as a number from **1** to **250**. Specifying the string **"unlimited"** sets no limit other than the size of memory available in the Annex. Alternatively, you can set the size to **"none,"** which forces the Annex to query the name server for each host name.

Minimum Uniqueness

Minimum uniqueness provides an ease-of-use feature, which allows users to enter only the characters necessary to uniquely match an entry in the host table. However, users can force the Annex to select only an exactly matching host name by enclosing the name they enter in double quotes. For example:

```
annex:  rlogin "widget"
```

If the host table contains the name *widgetslips*, and you want to log in to a host named *widget*, which is not in the host table, entering *widget* without the quotes causes the Annex to select *widgetslips*. Entering the name enclosed in double quotes forces the Annex to query a name server, because an exactly matching name is not in the host table. The minimum uniqueness feature can be turned off entirely by setting the **min_unique_hostnames** parameter to **N**.

Using Event Logging

The Annex provides the ability to log events to a 4.3BSD system log daemon (**syslogd**). The Annex may be able to log events to a 4.2BSD system using the **syslog** syslogging daemon or to a System V if it has system logging similar to 4.3BSD syslogging.

The 4.3BSD system logging daemon provides a *facility* as an addition to the *selector* field. The selector field is a list of priorities for a message and includes a level, which indicates the severity of a message. The facility defines the part of the system that generates the message. Certain facilities are reserved, such as kernel, mail, and daemons; other facilities can be defined in the configuration file **/etc/syslog.conf**. Facilities allow you to selectively log messages by priority.

Both the host and the Annex must be configured for system logging. To configure the Annex, **na** provides three parameters: **syslog_facility, syslog_mask**, and **syslog_host**. The **syslog_facility** parameter defines the facility used in the Annex syslog messages (specified as **log_local***n* where *n* is a number from 0 through 7). The default is **log_local7**. If the host to which messages are logged does not support 4.3BSD syslogging, this parameter is ignored and messages are logged only by priority level as defined by the **syslog_mask**. The **syslog_mask** parameter determines the priority levels that are logged. The possible values are **all, none**, or a combination of levels. The default, **none**, disables logging. The levels in priority order are:

emergency	Hardware failures.
alert	All Annex reboots.
critical	Configuration and initialization problems, such as format errors in the **gateways** file or lack of memory.
error	All line initialization errors, including CLI.
warning	Indications of minor problems.
notice	Time server queries and information about responses.
info	Starting and ending of CLIs and of Annex jobs created by the **rlogin** and **telnet** commands and the **ping** and **tap** superuser CLI commands.
debug	Activation and exit of all Annex processes.

When defining a priority level for a 4.3BSD syslog, all messages of that level or higher (that is, of greater severity) are selected. For example, if a level of **error** is selected with a 4.3BSD syslog, all messages for **error**, **critical**, **alert**, and **emergency** will be logged.

When defining a priority level for the Annex syslog, only messages for the level defined with this parameter are logged. Therefore, to log messages from a specific level or higher, you need to define all levels that apply. That is, if you want to log the same levels as shown above, you must specify all four levels. The level **all** has the same effect as specifying **debug** for a 4.3BSD syslog.

The **syslog_host** parameter defines the host configured to log Annex messages. The default, 0.0.0.0, causes the Annex to broadcast its log messages. Any syslog host on the network configured for the log message will log the message. That is, if the **syslog_facility** is configured, any 4.3BSD host configured for that facility logs the broadcast message.

Using the Time Server

The Annex maintains a UNIX-style time-of-day clock, which is based on the Internet date and time server, to regularly synchronize its clock. 4.3BSD implements an appropriate time server internally. The Annex always queries the preferred load host for the time.

The Annex distribution includes source code for a time server in case one is not available on the preferred load host. The Annex synchronizes its clock by requesting the time from a time server. The time server expresses time in the number of seconds since midnight (00:00:00), January 1, 1970, Greenwich Mean Time (GMT). The Annex converts time server time to local time and uses it to display with the CLI **stats** and **who** commands, to log events to **syslog**, and to calculate time of a boot and/or dump.

The Annex requests the time when it boots and synchronizes its clock with a server every 30 minutes. The Annex always queries the preferred load host first if one is defined. When the Annex is trying to set its starting time initially, it will default to the default starting time of 00:00:00, January 1, 1970, if a time server does not respond.

If a time server is not available on the preferred load host, the Annex, by default, does not broadcast for the time. However, you can enable broadcasting for a time server by setting the **time_broadcast** parameter to **Y**. Most UNIX systems provide a time server with the **inetd** daemon. Every host on the network that has a timer server will respond to a broadcast for the time.

The Annex does not reset its time during regular synchronization requests if a time server does not respond. It does not change its time by more than 10 minutes based on an answer to a broadcast request. That is, if the time returned to the broadcast query was greater than 10 minutes from the Annex's current time, the Annex only resets its time by a maximum of 10 minutes. If the timer server is on the preferred load host, the Annex adjusts to the time reported by the time server, regardless of the time interval.

The **timezone_minuteswest** parameter defines the time zone in which the Annex resides. Enter a positive number of minutes for time zones west of GMT and a negative number for time zones east of GMT. For example, since Eastern Standard Time is west of GMT, its value is 300 minutes; since Paris is east of GMT, its value is -60 minutes.

The **daylight_savings** parameter defines the daylight savings time to which your geographic area adheres. The Annex uses this parameter to adjust the time display for daylight savings time. You can specify daylight savings time as: **us, canadian, british, australian, west_european, east_european, mid_european,** or **none.**

Customizing the Annex Environment

You can customize the following Annex attributes:

- The prompt that displays when a user accesses the CLI.
- The total number of virtual CLI connections that can be created simultaneously at the Annex.
- The name of the message-of-the-day file.
- The RIP-listener.
- The type of IP encapsulation used by the LAN.

Setting the CLI Prompt

The Annex displays a prompt when a user accesses the CLI. The **cli_prompt** parameter allows you to customize this prompt for the entire Annex. Optionally, you can customize the prompt for each serial port using the **prompt** port parameter (see *Book D: Reference, Chapter 2: na Parameters*).

The values for this parameter are called prompt strings. A prompt string consists of characters and embedded formatting codes that are expanded when the prompt is displayed. The formatting codes consist of a percent character (%) followed by a single lower-case character. Each formatting code occupies one character in storage. You can also include a string with these codes for the prompt. The default prompt is **annex:**. Table B–1 lists and describes codes for the prompt string.

Table B–1. Formatting Codes for Annex Prompts

Code	Expansion
%a	The string **annex.**
%c	A colon followed by a space.
%d	The current date and time in standard UNIX format, such as Mon Mar 14 13:59:42 1989.
%i	The Annex's Internet address, such as 132.245.6.40.
%j	A new line character, skip to the beginning of the next line.
%l	The location defined for the port; if none, the string **port** *nn*,, where *nn* is the number of the serial line.
%n	The Annex's name or Internet address, such as 132.245.6.40.
%p	The port number or number for the virtual CLI connection in form of v*n*, where *n* is the number of virtual CLI connection.
%r	The string **port.**
%s	A space.
%t	The current time in 24-hour format, such as 13:59:42.
%u	The user name defined for the port; if none, a null string.

The following are examples of a prompt string and the CLI prompt that results from the string (a □ represents a space). If you want a prompt to appear as *username*□*location*:□, use the following code:

%u%s%l%c

If the user name is defined as *abercrombie* and the location as *lobby* for the port, the prompt is:

```
abercrombie□lobby:
```

If neither are defined on port 3, the prompt for port 3 is:

```
□port□3:
```

If you want a prompt to appear as *date and time* (new line) *annex-name*□*port number*:□, use the following code:

%d%j%n%s%p%c

For the port on the Annex named *thirdfloor*, the prompt for port 6 is:

```
Mon May 16 11:10:25 1989
thirdfloor□6:□
```

For the superuser CLI prompt, a pound sign (#) and a space replace the code **%c**; otherwise a # is appended at the end.

Setting a Limit on Virtual CLI Connections

The number of virtual CLI connections at an Annex can affect the use of memory, as each virtual CLI connection uses memory. The **max_vcli** parameter determines the maximum number of virtual CLI connections the Annex can create at any one time. You can set the number of virtual CLI connections from an unlimited number to none. The range of values that you can enter are from **0** to **254** or **"unlimited."** The default is **"unlimited."** If you define this parameter as zero, users cannot create a virtual CLI connection at the Annex.

Setting the Message-of-the-day File

The Annex can display a message-of-the-day at the CLI prompt after it has been rebooted or reset, or the port has been reset, or each time a user accesses the Annex through a virtual CLI connection.

The message-of-the-day resides in an ASCII file located in the **/usr/spool/erpcd/bfs** directory on a file server host. The default file name is **motd**. The **motd_file** parameter allows you to specify another name for this file. The Annex reads this host file each time it is booted, and when the **na** command **reset annex motd** is issued. The Annex first requests the file from the preferred load host. If it does not receive the file, it broadcasts its request unless the **load_broadcast** parameter is set to **N**.

Using the RIP-listener

The Annex uses a routing daemon similar to the BSD routing daemon (**routed**) for its routing services. This daemon uses Routing Information Protocol (RIP) messages to create routing tables. RIP messages are sent by a gateway announcing that it provides the route to a destination.

The **routed** parameter defines whether or not the routing daemon's (**routed**) RIP-listener is enabled or disabled. If the RIP-listener is disabled, the Annex does not listen for RIP routing broadcasts to update its routing tables. The Annex depends only on the routing information provided in its **gateways** file, which was downloaded during the boot. However, the Annex will update its routing table in response to ICMP redirects (see *Chapter 7: Configuring Hosts and Servers* in this book). The default is RIP-listener **enabled.**

Setting the IP Encapsulation Type

The Annex supports two type of LAN encapsulation of IP packets: Ethernet Version 2 format or the IEEE 802.2/802.3 Data Link Layer format. The **ipencap_type** parameter specifies which type of encapsulates to use; the default is **ethernet.**

This parameter can be changed only at installation time using the ROM Monitor because the Annex cannot boot without the correct IP encapsulation. Do not change this parameter using **na** because the Annex will not be able to boot.

Note: Original Annexes do not support IEEE 802.2./802.3 IP encapsulation.

Setting for a Token Ring Network

The Annex software supports IEEE 802.5 and IBM Token Ring network. This communications server allows TCP/IP connections between attached devices and hosts on the token ring network.

Only the **ring_priority** parameter is used to configure the Annex running IEEE 802.5. This parameter sets the priority for Annex-originated messages on the token ring LAN. The values that can be entered are **0** to **3**. The default is **0.**

Note: The Annex does not support media access control level source routing.

Chapter 2

Configuring Ports

General

With an Annex you can attach terminals, printers, modems, PCs, and hosts lacking a network interface to a serial line port. The Annex supports the Serial Line Internet Protocol, which allows you to attach remote local area networks (e.g., Ethernet), remote Annexes, remote PCs, etc. This chapter discusses:

- General issues on configuring ports
- Configuring ports for terminals
- Configuring ports for hosts

General Issues on Configuring Ports

The port parameters you must set vary based on the device your are attaching to the port. However, discussions on setting some port parameters are similar for many devices. These discussions include:

- Using **na** to set port parameters
- The **mode** parameter
- Port security parameters

Configuring Ports Using the na Program

All parameters, except the Annex's Internet address, have default values. When configuring ports, modify only the parameters whose defaults differ from your requirements. The **na** program provides two commands for setting port parameters: **set port** and **set printer**.

The **set port** command sets parameters for the serial line ports. The **set printer** command sets parameters for the parallel printer port to which you can attach a printer using a Centronics interface or, as an option with the Annex IIe or Annex 3, a Dataproducts interface.

Also, **na** provides two commands for displaying the current port settings: **show port** and **show printer**. The **show port** command provides five keywords:

- **all**: displays all port parameters
- **interface**: displays the device's interface attributes
- **device**: displays the device's attributes
- **editing**: displays the CLI line editing parameters used with terminals
- **slip**: displays the port's SLIP parameters

To configure an Annex serial port:

1. Execute the **na** program as follows:

```
% na
Annex network administrator R6.0.2 May 3, 1991
command:
```

2. Specify an Annex. For example:

```
command: annex 132.245.6.40
```

   Or specify multiple Annexes. For example:

```
command: annex 132.245.6.40, front lobby
```

   or

```
command: annex
  enter default annex 132.245.6.40, front lobby
  list:
```

3. Specify a port. For example:

```
command: port 1
```

   Or specify multiple ports. For example:

```
command: port 1-10
```

Note: You can skip step 2 by specifying the Annex with an @ following the port number(s). For example:

```
command: port 1@132.245.6.40
```

or

```
command: port 1-10@132.245.6.40
```

4. Execute the **set port** command along with those parameters you want to change. The following example shows how to configure a port for a VT100- compatible terminal:

```
command: port 8
command: set port data_bits 7 parity even user robert\
         inactivity_timer 30 location lab\
         term_var vt100 cli_security Y\
         connect_security Y port_password secret
```

5. Execute the **show port** command to review your changes. Using the above example, the terminal would display:

```
command: show port

annex rabbit port 8:
                speed: 9600
            data_bits: 7
            stop_bits: 1
               parity: even
          imask_7bits: N
        control_lines: none
                 type: hardwired
                 mode: cli
    forwarding_timer: 0
      cli_inactivity: off
    inactivity_timer: 30
     allow_broadcast: Y
broadcast_direction: network
   max_session_count: 3
 input_flow_control: bell
    input_start_char: ^Q
     input_stop_char: ^S
output_flow_control: start/stop
  output_start_char: ^Q
   output_stop_char: ^S
 ixany_flow_control: N
```

```
            long_break: Y
           short_break: Y
             attn_char: ^@
             user_name: "robert"
              term_var: "vt100"
     input_buffer_size: 1
  bidirectional_modem: N
 default_modem_hangup: N
     forwarding_count: 0
              location: "lab"
    input_is_activity: Y
   output_is_activity: N
   reset_idle_time_on: input
         cli_security: Y
     connect_security: Y
 port_server_security: N
        port_password: "<set>"
    dedicated_address: 0.0.0.0
       dedicated_port: telnet
               prompt: ""
     newline_terminal: N
     leap_protocol_on: N
                 echo: Y
          map_to_lower: N
          map_to_upper: N
            char_erase: Y
            line_erase: Y
         hardware_tabs: Y
            erase_char: ^?
            erase_word: ^W
            erase_line: ^U
       redisplay_line: ^R
        toggle_output: ^O
        telnet_escape: ^]
   slip_local_address: 0.0.0.0
  slip_remote_address: 0.0.0.0
     slip_subnet_mask: 0.0.0.0
  slip_load_dump_host: 0.0.0.0
          slip_metric: 1
      slip_allow_dump: Y
   slip_do_compression: N
slip_allow_compression: N
         slip_no_icmp: N
             slip_tos: N
```

6. Execute either the **boot** or **reset** *port* command to effect these changes at the Annex.

You can configure multiple ports on multiple Annexes with a few simple steps:

1. Define a port with the **port** command.
2. Define the parameters for that port.
3. Use the **copy port** command to copy those parameters to other ports on one or more Annexes.

The following example illustrates the above steps:

```
command: port 1
command: set port speed 19,200 databits
command: copy port 1@132.245.6.40 2-8,12,14,15,16
```

Alternately, you can define all parameters for one Annex including port parameters. Next, use the **write** command to create a script file with all configuration data for that Annex. Finally, execute the **read** command for all Annexes you want to configure.

Note: The **write** command does not write the port password.

Port Mode

An Annex port can be opened from a device attached to the port or from the network requesting attachment to a device. The port mode, specified by the **mode** parameter, dictates the direction from which the port can be opened. The port mode options are **cli, dedicated, slave, adaptive, slip,** and **unused.**

Both **cli** and **dedicated** mean that the port always is opened from the device. The difference between these two modes is that **cli** allows access to the Annex's Command Line Interrupter (CLI) providing access to multiple hosts. A port defined as **dedicated** can access only one host using one connection request command (either **rlogin** or **telnet**). Typically, a terminal or an inbound modem is attached to a port defined as either **cli** or **dedicated.**

A port defined as **slave** can be opened only by a request from the network. This access is provided through the Annex's port server, which accepts Telnet connection requests from users or applications on the network. The port server frequently is used for printers, outbound modems, virtual CLIs, and ASCII hosts lacking a network interface (see *Chapter 3: The Port Server and Rotaries* and *Chapter 4: Printer* in this book).

A port defined as **adaptive** is used for bidirectional modems. An **adaptive** port becomes a **cli** port when an incoming call is received and a **slave** port when the port receives a request from the network to make an outgoing call. Chapter 5 in this book, *Modems*, provides a full discussion on configuring a port for modems.

A port defined as **slip** is used to attach another host, a PC, another Annex, or any device that supports SLIP (see *Chapter 6: Serial Line Internet Protocol (SLIP)* in this book).

The Annex ignores a port defined as **unused**.

Port Security

The Annex provides a security system that allows you to configure security on a per-port basis. You can use host-based security, local password protection, or a combination of the two.

Host-based security provides access validation for the following access requests:

- From the device, a terminal or modem, attached to the port
- From a user on the network connecting to the port
- From a device attached to the port connecting to a host or a network

Local password protection can be used as a stand-alone security mechanism or as a backup to host-based security. It provides validation for accessing a port from either the device or the network. Local password protection supports ports defined as CLI, slave, or adaptive.

The host-based security also allows you to mask out CLI commands. When a command is masked, it is not displayed with the **help** command; and if entered, CLI displays an error message. You can customize the security policy to meet your individual security requirements. The local password protection policy cannot be altered. (see *Chapter 1: Configuring the Annex*, *Chapter 3: The Port Server and Rotaries*, and *Chapter 7: Configuring Hosts and Servers* in this book for more information).

Note: For any security mechanism, you must enable security for the Annex by setting the Annex **enable_security** parameter to **Y**.

Validating CLI Connections

Both host-based port security and local port password protection provide security validation on CLI connections from attached devices. The Annex requires user validation when the port is opened. With host-based security, the user validation policy requests a user name and password. Local password protection requires only a password.

Note: Original Annexes do not support local password protection.

To use host-based security for CLI security:

- Set the **cli_security** port parameter to **Y**.
- Create a password file on the security server(s). The Annex's security system (ACP) expects the file to be **acp_passwd** (see *Chapter 7: Configuring Hosts and Servers* in this book).

To use local password protection as a back-up to the host-based security, define a password using the **port_password** parameter. To use only local password protection for CLI security, set the **cli_security** parameter to **N** and define a password with the **port_password** parameter. CLI connection security is applicable to terminals and inbound and bidirectional modems. If the port password is used as a back-up to the host-based CLI security, and the security server is unavailable, the console terminal displays a status message when a user tries to access a port:

```
Checking authorization, Please wait...
Remote security server timed-out, invoking local security.
Port password:
```

With host-based security, users can change their passwords using the **ch_passwd** utility (see *Book D: Reference, Chapter 4: Utilities*).

Connection Security

Host-based security provides connection security that can restrict access to designated hosts and networks. Connection security is applicable to terminals and to inbound and bidirectional modems.

To implement connection security:

- Set the **connect_security** parameter to **Y**.
- Create a restriction file on the security server(s). ACP expects the file to be **acp_restrict** (see *Chapter 7: Configuring Hosts and Servers* in this book).

The **acp_restrict** file lists those hosts and networks to which access from the specified Annex is restricted. When a user requests a connection to a host, the Annex verifies the ability to connect to that host (or network). If the host is listed as restricted, access to that host is denied for any port on which connect security is enabled.

For virtual CLI connections, connection security is enforced for all restricted hosts: if a host is listed as restricted for an Annex, all virtual CLI connections are denied access to that host.

Configuring Ports for Terminals

Terminals are the devices most frequently attached to a communications server. The Annex allows three port configurations for a terminal: **cli, dedicated,** and **slave.**

CLI Ports

A CLI port has the **mode** parameter set to **cli.** In the simplest configuration, the CLI prompt displays when the user turns on the connected terminal or presses the **Return** key. At this point, the user has access to all permissible CLI commands. You can limit the number of sessions the user can initiate, and you can configure several different security options. When configuring CLI port parameters:

- Configure the **speed,** the **data_bits,** the **stop_bits,** and the **parity** parameters to match the terminal settings.
- Set the **type** parameter to **hardwired.**
- The **max_session_count** allows you to limit the number of sessions a user can activate simultaneously. Setting the value to one limits the user to one session at a time. The default is three (with a maximum of 16).

- Set the **allow_broadcast** parameter to **N** if you want to disable the display of administrative messages generated with the **na broadcast** command at the port.
- The **user_name** and **location** parameters part used for administrative information. The CLI **who** command displays this information. Also, the **user_name** is passed in the **rlogin** command's connection request. If the user does not have an account on the host under the same user name you defined with the **user_name** parameter, the user must issue the **rlogin -l** command.
- The **term_var** parameter is a string that identifies the terminal type. Any value defined for this parameter is passed with both the **telnet** and **rlogin** connection requests. If you define a terminal type, it must be one that is valid for the host to which the user is connecting.
- CLI activity timers provide simple security by resetting unattended ports. Limited resources, like dial-in modems, are released when not in use (see *Port Security*).

  The **inactivity_timer** specifies an amount of time in minutes that the port can be inactive before the Annex hangs up the port. When this timer expires, all sessions are terminated and the port is reset. Allowable values for this parameter are **0** to **255**. The default is **0** (displays as **off**).

  Activity can be set to either input (data received from the terminal), output (data sent to the terminal), or both. Set the **input_is_activity** parameter to **Y** and/or the **output_is_activity** parameter to **Y**.

  For a terminal connected to a CLI port, only the **input_is_activity** parameter needs to be set. This resets the inactivity timer each time the user types at the terminal. Setting the **output_is_activity** parameter resets the inactivity timer when the user receives messages at the terminal; for example, messages announcing mail has arrived or a batch job has completed.

  The **cli_inactivity** timer specifies the amount of time that a CLI port with no active sessions can remain inactive before the Annex resets the port. Allowable values for this parameter are **0** to **255**, and **immediate**. The **immediate** value directs the Annex to reset the port immediately after the last session closes. The default is **0**.

- An attention key, when pressed, notifies the Annex that the user wishes to suspend an ongoing session with a host and return to the CLI to enter commands. The Annex provides three parameters for defining attention keys: **short_break** enables the **Break** key on many terminals; **long_break**enables a key that generates a long break; and **attn_char** defines a key sequence that produces a break.

Generally, the attention key defined with the **attn_char** parameter is used with virtual CLI connections to an Annex, in which case the default is CTRL-A. You may want to suggest to your users that they define their own attention character with the CLI **stty attn** command. The attention key can also program a function key to transmit an interrupt.

- Typically, parameters that display with the **show port editing** command define characters that provide CLI line editing functions. Some of these characters are passed as Telnet special characters with CLI-connected terminals. The CLI connected terminal is the only terminal that uses these parameters.

Dedicated Ports

A port set to **mode dedicated** has access to a single host. Connection to the host is made using either the **rlogin** or **telnet** protocol. Using **telnet**, you can configure a TCP port to allow the automatic start-up of an application on the host.

A **dedicated** port has two mechanisms for originating the connection, depending on the setting of the **control_lines** parameter. If **control_lines** is set to **modem_control** or **both**, a connection is made automatically when the Annex sees DCD asserted. If **modem_control** is not in use, a **Return** initiates the connection.

- Set the **speed, data_bits, stop_bits,** and **parity** parameters to match the terminal's settings.
- If the **dedicated** port is connected directly to a terminal, set the **type** parameter to **hardwired**. If the connection times out without having received input from the user, the terminal displays the following message: *Press return to restart login*. The Annex then waits for input, or for the terminal to be turned off and turned back on. In either case, the Annex restarts the login.

  If the **dedicated** port is connected to a modem, set the **type** parameter to **dial_in**.
- Set the **dedicated_address** parameter to the target host's IP address.
- Set the **dedicated_port** parameter to **telnet, rlogin,** or a specific TCP port number. The target host ***must*** have a daemon listening on the specified TCP port.
- Setting the **allow_broadcast** parameter permits the display of any administrative messages at the terminal connected to the dedicated port.
- Set both the **user_name** and **location** parameters; the CLI **who** command displays these parameters. If the **dedicated_port** parameter is set to **rlogin**, the user name is passed with the connection request.
- The **term_var** parameter's value is passed with either connection request.
- If you set the **inactivity_timer** parameter, also set the **input_is_activity** or **output_is_activity** parameters (or both).

- The **telnet_escape** character allows the user to escape to the **telnet** prompt. Setting the parameter to **U** disables the Telnet escape.

Note: You cannot implement either host-based security or local password protection for dedicated ports.

Slave Ports

A port set to **mode slave** accepts TCP connections over the network. Slave ports are used primarily for modems, printers, and other serial devices. A possible application allows a **getty** process to access a terminal attached to the port. This application requires the Annex **rtelnet** utility to provide a device file for the **getty** process.

- Set the **speed, data_bits, stop_bits,** and **parity** parameters to match the terminal's settings.
- Set the **type** parameter to **hardwired.**
- Setting the **allow_broadcast** parameter to **Y,** and the **broadcast_direction** parameter to **port,** allows a user at the terminal to receive any administrative broadcasts.
- The CLI **who** command displays the **user_name** and **location** parameters.

Host-based Applications to Access a Terminal

The Annex **rtelnet** daemon, which is included with the Annex software, provides Telnet connections between a port and a character device on a host. This connection opens the port to allow input from or output to the terminal. If you configure **rtelnet** to start when the host boots, it will provide the link between a **/dev** file on the host and the port. The following example sets up a **getty** line as shown in Figure B–1; the terminal is in a lobby and is connected to port 3 on *annex02*.

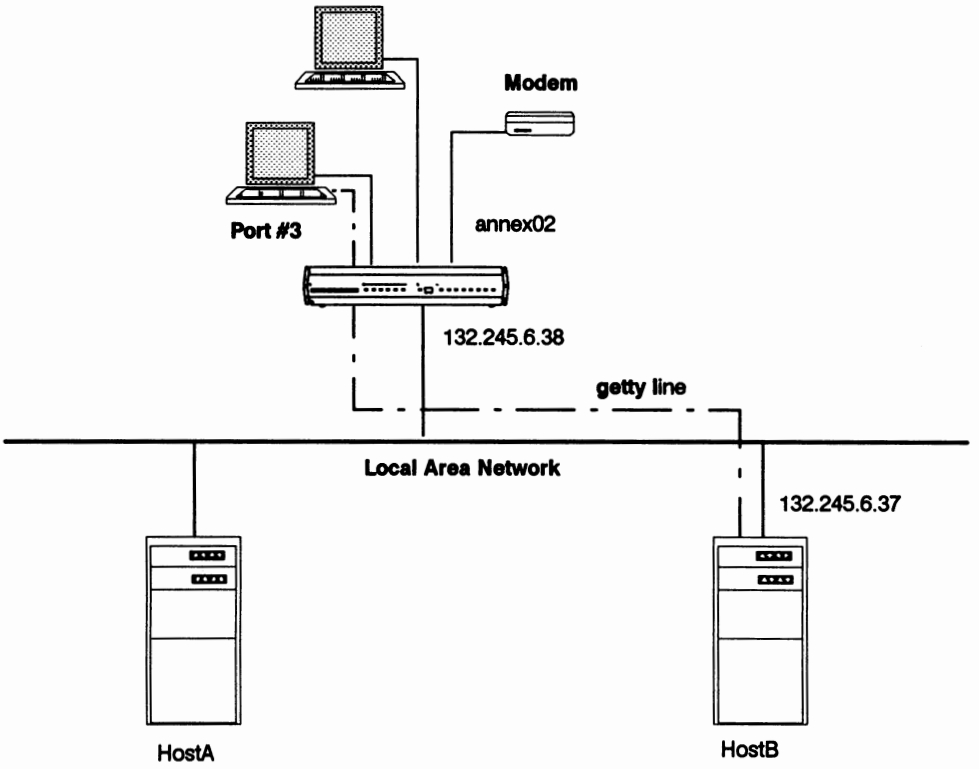

Figure B–1. Host Applications Accessing a Terminal

The steps involved in creating this example for a BSD UNIX host are:

1. Create a special file with **rtelnet** as follows:

   ```
   rtelnet -r annex02 3 /dev/ttyDB
   ```

   You can specify the Annex by either its Internet address or its name. If you use the name, make sure that it is listed in the name server database and that the name server is started before the **rtelnet** command.

2. Add a line to **/etc/ttys** to define **/dev/ttyDB** as a **getty** line:

   ```
   ttyDB "/etc/getty std.9600" ansi on
   ```

3. Signal **init** so that it reads **/etc/ttys** and starts a **getty** now:

```
# kill -HUP 1
```

4. Add the **rtelnet** command to **/etc/rc** so that the special file is created when the system is booted.

The steps involved in creating this example for a System V host are:

1. Create a special file with **rtelnet** as follows:

```
rtelnet -r annex02 3 /dev/ttyDB
```

   You can specify the Annex by either its Internet address or its name.

2. Add a line to **/etc/inittab** to define **/dev/ttyDB** as a **getty** line:

```
DB:2:respawn:/etc/getty ttyDB 9600
```

3. Signal **init** so that it reads **/etc/inittab** and starts a **getty** now:

```
# /etc/telinit q
```

4. Add the **rtelnet** command to the appropriate **/etc/rc** so that the special file is created when the system is booted.

Configuring Ports for Hosts

The Annex provides a front-end service to a host that does not have a network interface by attaching the Annex's ports to the host's serial ports.

- Set the **mode** parameter to **slave**.
- Set the **speed**, **data_bits**, **stop_bits**, and **parity** parameters to match the requirements of the host's serial lines.
- Set the **imask_7bits** parameter to **Y** so that the Annex ignores the received parity bit. Some UNIX hosts transmit different parity depending on whether **tty** is in raw or cooked mode.

- Set the **type** parameter to **hardwired.**
- If you set the **allow_broadcast** parameter to **Y**, setting the **broadcast_direction** parameter to **network** allows the users on the network connecting to the host to receive any administrative broadcasts.
- For port server security, either set the **port_server_security** parameter to **Y** or define a password for the **port_password** parameter or both. Also, set the Annex **enable_security** parameter to **Y.**
- The CLI **who** command displays the **user_name** and **location** parameters.

Chapter 3

The Port Server and Rotaries

General

As a port server, the Annex accepts TCP Telnet connection requests from users and applications on the network. Once connected to the Annex, a user or application can connect to a device attached to a port that is in slave or adaptive mode. The following example shows how a user connects to an Annex using **telnet** (see Figure B–2):

```
% telnet annex01
Trying...
Connected to annex01.
Escape character is '^]'.

Rotaries Defined:
   cli
   Enter Annex port name or number: 2
   Attached to port 2
```

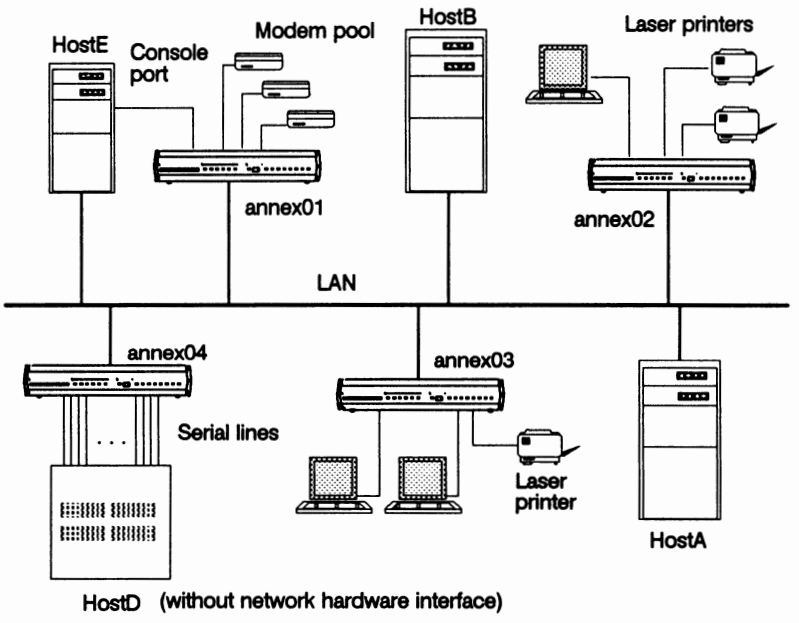

Figure B–2. Connecting Devices to an Annex

The Annex named *annex01* provides modems for outgoing and incoming calls. The Annexes named *annex02* and *annex03* provide laser printers to which printing applications can be sent. The Annex named *annex04* provides access to a host lacking a network interface. The Annex named *annex01* provides the console port for remote system management on hosts.

The Port Server

Once connected to an Annex by Telnet, the port server provides users with the option of selecting a single port, a rotary, or a CLI connection.

Applications connecting to an Annex access only a single port.

A rotary is a set of Annex ports grouped together so that they can be addressed by users, and managed by the Annex, as a single entity.

You can create rotaries by defining them in an ASCII file. This file must be compiled in order to download it to an Annex. Defining rotaries in a file allows using the name instead of a range of numbers when selecting a port. The syntax rules and how to compile this file are described later in this chapter.

If the user selects a CLI, the Annex creates a virtual CLI connection. A virtual CLI permits access to CLI commands from anywhere on the network. This provides administrators with remote system management capability.

When a user requests a rotary by entering a range of numbers, the port server attaches the user to the first available port. However, if the requested port is busy or all ports in a requested rotary are busy, the port server asks if the user wants to wait. This feature is called camp-on. If the user chooses to wait, the Annex puts the request in a first-come, first-served queue and notifies the user when a port is free.

Using the Annex, you can set up security for both the port server and the virtual CLI connections.

Also, the Annex supports ranges of TCP port numbers that, when entered with the **telnet** command, are mapped directly to a port or to a defined rotary.

Camp-on

Camp-on is provided when all ports in a rotary are busy or when the specified single port is busy. The following example uses Figure B–2 to illustrate the camp-on feature. A user at a terminal connected to *annex02* logged into *HostB* using **rlogin** and wants access to a modem connected to port 6 on *annex01*. At the CLI, the user issues the **telnet** command to connect to *annex01*:

```
 annex: jobs
+1  rlogin HostB
 annex: telnet annex01
Trying...
Connected to annex01.
Escape character is '^]'.

Rotaries Defined:

     cli

Enter Annex port name or number:  6
Port(s) busy, do you wish to wait? (y/n) [y]: y
```

The *yes* response places the request in a first come, first served queue. At this point, the user can return to an existing job or create another job. If the user places the connection request into the background, the Annex notifies the user when the connection is complete. While waiting for this notification, the user can escape to the **telnet** prompt.

In the next example, the user issues the CLI **bg** and **fg** commands to return to *HostB*.

```
annex: bg
2   telnet annex01 &
annex: jobs
+1  rlogin HostB
-2  telnet annex01 &
annex:  fg 1
rlogin HostB

%
```

Note: The Annex cannot send messages to the CLI from jobs placed in the background. To display messages from a background job, you must be in a session to another host.

The user continues working on *HostB* until notified that the port is free. The terminal displays the following sample message:

```
Attached to port 6
```

The user can return to the CLI and issue the **fg** command to access the Telnet session. At this point, the user is connected to the modem and can use a dial command. For example:

```
annex: jobs
+1  rlogin HostB
-2  telnet annex01 &
annex: fg 2
 2   telnet annex01
ATD ...
```

If you create rotaries in the rotary file, you can force camp-on without questioning the user or disable camp-on entirely (see *Rotaries* in this chapter).

Annex-specific TCP Port Numbers

Telnet can include a TCP port number in the connection request. The Annex recognizes the following ranges of TCP port numbers: 5000, 6000, and 7000.

The port numbers in both the 5000 and 7000 range map directly to serial ports: TCP port 5001 maps to serial port 1, TCP port 5002 to serial port 2, TCP port 7001 to serial port 1, etc. (you do not need to configure these directly-mapped TCP ports).

TCP ports in the 5000 range use the Telnet protocol. Connecting to a TCP port in this range attaches directly to the specific serial port. The result is the same as selecting the serial port through the port server, but there is no negotiation or camp-on. The connection is refused if the serial port is busy or not in slave mode.

From the previous example, the user could have requested TCP port number 5006 using the **telnet** command. For example:

```
annex: telnet annex01 5006
Trying ...
    Connected to annex03.
    Escape character is '^]'.

    Attached to port 6
```

TCP port numbers in the 7000 range also provide a direct-mapped connection, but do not use the Telnet protocol. These TCP ports are similar to raw connections in that they pass data directly to and from the serial port.

TCP port numbers in the 6000 range are used when defining rotaries in a file (i.e., a TCP port number in the 6000 range can be assigned to a single rotary). Issuing a Telnet connection request to such a TCP port would result in directly attaching to the rotary (see *Rotaries* in this chapter).

Virtual CLI Connections

The Annex provides the ability to access the CLI from anywhere on the network through the port server. The Annex creates a virtual CLI connection for the user when either a CLI is requested at the port server prompt or the TCP port number 5000 was included in the **telnet** command.

A virtual CLI connection generally behaves the same as CLI connected terminal. That is, with a virtual CLI a user can access any CLI commands not excluded through security, including connection requests. However, on a virtual CLI connection, most port characteristics as displayed with the CLI **stty** command have no effect on the connection with the exception of the attention character.

The attention character is defined by **stty** as **attn**. The default is CTRL-A and is only used with virtual CLI connections. The attention character provides the user with access to the CLI while in a session with another host.

The Annex creates a new virtual CLI connection for each request it receives. You can limit the number of virtual CLI connections the Annex creates using the **max_vcli** Annex parameter. The only other limit on the number of virtual CLI connections created is system resources.

Security for the Port Server

The port server security provides the option of configuring host-based security, local password protection, or both. The host-based security for the port server normally requires a user name and password. Local password protection on the port server requires only a password.

As with security on CLI connections, the local password protection can be used as the sole security mechanism or as a backup to host-based security for those occasions when the security servers are not available.

With port server security, the port server invokes the security mechanism when the user requests access to a specific port or rotary at the port server prompt. User validation occurs before the user is connected to the port to ensure that the user is authorized to connect to the selected port. If the user is not authorized, the port server notifies the user and prompts for another port. Below is an example of the supplied user validation for port server security.

```
annex: telnet annex01
Trying ...
    Connected to annex01.
    Escape character is '^]'.
    Rotaries Defined:
    modem_1200        1-3,5
    modem_2400        4,7
    cli               -

 Enter Annex port name or number:modem_1200
 Annex username: kathryn
  Annex password:

  Permission granted
  Attached to port 1
```

For camp-on, user validation occurs when a port becomes available. If the name or password is incorrect, the user is returned to the port identification prompt.

To use host-based security with the port server:

- Enable security for the Annex; set the **enable_security** Annex parameter to **Y**.
- Set the **port_server_security** port parameter to **Y**.
- Create a password file on the security server(s). ACP expects the file to be **acp_passwd** (see *Chapter 7: Configuring Hosts and Servers* in this book).

For local password protection with the port server, define a password for the **port_password** port parameter whether you use port password as the only security or as a backup for host-based security.

Note: The **port_password** port parameter is applicable to both CLI and port server connections.

Security on Virtual CLI Connections

The Annex establishes security on virtual CLI connections using host-based security, local password protection, or both. Host-based security uses a user name and password validation.

The virtual CLI security mechanism is similar to the port server security mechanism in that user validation is invoked after the user has requested access to the CLI at the port server prompt. This ensures that the user is authorized to access the CLI.

To set up host-based security on virtual CLI connections:

- Enable security for the Annex: set the **enable_security** Annex parameter to **Y**.
- Enable virtual CLI security: set the **vcli_security** Annex parameter to **Y**.
- Create a password file (**acp_passwd**) on the security server.

Note: Setting **vcli_security** to **Y** has the same effect as enabling **cli_security** and **connect_security** on a serial port.

Local password protection requires only a password for validation. To set up this protection:

- Enable security for the Annex: set the **enable_security** Annex parameter to **Y**.
- Disable virtual CLI security: set the **vcli_security** Annex parameter to **N**.
- Define a password for all virtual CLI connections for the Annex using the **vcli_password** Annex parameter.

The local password protection can function as a back-up security system to host-based security. To do this:

- Set up host-based security for the virtual CLI connection.
- Define a virtual CLI connection password using the **vcli_password** Annex parameter. Virtual CLI connections must adhere to any connect security defined for the Annex.

Rotaries

A rotary is a group of serial ports that the Annex manages as a single entity. A rotary exists any time a group of ports is requested at the port server. You can customize the behavior and use of rotaries by administratively defining rotaries in a file. Administratively defined rotaries can comprise one or more ports on one or more Annexes. You define rotaries in an ASCII file called **rotaries**. This file is downloaded onto Annexes.

When customizing rotaries, you can:

- Name rotaries
- Group rotaries from multiple Annexes together
- Assign Internet addresses
- Assign TCP port numbers
- Create invisibility
- Modify camp-on
- Create raw rotaries
- Force a binary mode connection

Rotary File

Each entry in the **rotaries** file consists of a rotary name and a definition. The simplest form for an entry in this file is:

```
HostC: 1-4,29-32@annex04
```

This example defines a rotary with the name of *HostC*. This rotary as illustrated in Figure B–2 uses ports 1 through 4 and ports 29 through 32 on *annex04*.

The following is an example of what the user sees accessing the port server on *annex04* through a **telnet** command:

```
% telnet annex04
Trying...
    Connected to annex04.
    Escape character is '^]'.


    Rotaries Defined:
    HostC               1-4,29-32
    cli                 -

Enter Annex port name or number:
```

Users can enter the rotary name or one or more of the port numbers to connect to *HostC* attached to the ports. Minimum uniqueness applies to rotary names; in this case, users can enter *HostC*, *Hos*, or *H*.

Rotaries File Syntax

Entries in the rotaries file follow these conventions:

- Lines beginning with a pound sign (#) are comments.
- Blank lines are ignored.
- A maximum of 132 characters per line.
- Entries can be continued on the next line by preceding the new line with a backslash (\).
- The following characters have special meanings: colon (:), plus-sign (+), *at* sign (@), slash (/), comma (,), semi-colon (;), backslash (\), and pound sign (#).
- The rotary name string cannot contain a space, a tab, or a comma (,) but it can contain the syntax characters listed above if they are escaped with a backslash (\).
- Spaces and tabs can be used anywhere to improve readability.
- Keywords must be delimited by spaces or tabs.

The syntax of each entry is:

rotary_name: [*keyword*] *ports@location*[;*ports@location*] ...

The *rotary_name* is the name of the rotary and ***must*** be followed by a colon (:). The maximum length is 32 characters.

The *ports@location* specifies the serial line ports and Annex(es) on which the rotary resides. The ports can be specified as:

- A single number. For example:
  `1@annex04` or `6@132.245.6.7`
- A list of numbers separated by commas. For example:
  `1,3,6,7@annex03` or `12,16,28@132.24.5.6.9`
- A range of numbers separated by a dash. For example:
  `6-9@annex01` or `2-12@132.245.6.5`
- Any combination of the above. For example:
  `1,3,7,6-9@annex01` or `2-12,16,23,29@132.245.6.5`

The Annex defined in the *location* argument must be preceded by an *at* sign (@) and can be specified using either the Internet address or a name specified in a name server database. The *location* argument supports the following two options:

@inet_addr+inet_addr	Defines an auxiliary Internet address that can also be used to access the rotary.
@inet_addr/tcp_port	Defines a TCP port number that can also be used to access the rotary. The number must be from 6000 through 6999.

The supported *keywords* are:

ps_visible **ps_invisible**	Controls displaying the rotary name at the port server. Only rotaries with an auxiliary Internet addresses or auxiliary TCP port numbers can be defined as invisible (**ps_invisible**). The default is **ps_visible**, except on raw rotaries which are always invisible.
telnet **raw** **binary**	Defines the protocol between the port and the device. The default is **telnet. raw** provides a data stream with no character processing; it is intended primarily for program access to the rotary. **binary** causes the Annex to negotiate with the host to operate in **telnet** binary mode in both directions.
direct_camp_on =	**ask** | **always** | **never** Defines how camp-on is handled. Typically, the default is **ask; raw** rotaries default to **never.**

Configuring Rotaries

Configuring rotaries includes the following options:

- Defining multiple rotaries with one file entry
- Assigning rotaries auxiliary Internet addresses
- Using the DNS server to define multiple rotaries
- Assigning rotaries TCP port numbers
- Configuring visibility
- Configuring camp-on
- Configuring raw rotaries
- Configuring binary rotaries

Defining Multiple Rotaries with One Entry

You can include more than one Annex in a single file entry by separating the *ports@locations* field with semicolons. The following entry defines a rotary named *modems* that resides on two different Annexes. The rotary on *annex01* has seven ports; the rotary on the Annex with the Internet address 132.245.6.15 has one port.

```
modems: 1,4,7,13-16@annex01; 10@132.245.6.15
```

When the user accesses *annex01* using a **telnet** command, the port server displays:

```
 % telnet annex01
Trying...
    Connected to annex01.
    Escape character is '^]'.


    Rotaries Defined:
    modems              1,4,7,13-16
    cli                 -

Enter Annex port name or number:
```

When the user accesses the Annex at 132.245.6.15, the port server displays:

```
% telnet 132.245.6.15
Trying...
    Connected to 132.245.6.15.
    Escape character is '^]'.

    Rotaries Defined:
    modems              10
    cli                 -

    Enter Annex port name or number:
```

Assigning Rotaries Internet Addresses

An auxiliary address allows you to assign an Internet address that connects directly to the rotary. The user can access the rotary by entering the auxiliary address using the **telnet** command. The auxiliary address must adhere to the standard network addressing conventions of your network.

Using auxiliary addresses to access a rotary changes the behavior of the port server. The port server does not display rotary names; instead, the port server attaches to the first available port. For example:

```
modems: 1,4,6-9@annex01+132.245.6.80
```

The user can issue the following command to access the first available port in the rotary. For example:

```
% telnet 132.245.6.80
Trying...
    Connected to annex01
    Escape character is '^]'.
    Attached to port 4
```

You can also add the rotary name and the auxiliary address to **/etc/hosts** or to the host name database so users can access the rotary directly by name. For example:

```
132.245.6.80            modems
```

The user can use the name *modems* and access the first available port in the rotary. For example:

```
% telnet modems
Trying...
    Connected to annex01
    Escape character is '^]'.
    Attached to port 7
```

Using the DNS Server to Define Multiple Rotaries

If you are using a Domain Name Service (DNS) server on the network, you can create an entry with multiple rotaries as described above, assign Internet addresses to these rotaries, and create entries in the name servers database for the name of the rotary. This allows users to request the rotary name with the **telnet** command.

With the DNS server, the Telnet request searches for the first available port by attempting to access the first port on the first Annex listed with the rotary. If that port is busy, it attempts to connect to the next port in the list, and so on until a port is available.

Using the following example, one entry defines rotaries on two Annexes:

```
modems: direct_camp_on=never\
        1,3,8,11@annex01+132.245.6.90;\
        6-8@annex05+132.245.6.91
```

In the DNS server's database, create two entries for the name *modems* with two different Internet addresses. For example, using a BIND name server:

```
modems      IN      A       132.245.6.90
            IN      A       132.245.6.91
```

When the user attempts to connect to *modems* using the **telnet** command, Telnet tries to locate an available port. First, it tries port 1 on *annex01*, next port 3, followed by port 8, and so on until an available port is located. For example:

```
% telnet modems
Trying...
    Connected to annex05
    Escape character is '^]'.
    Attached to port 6
```

Note: In the above example, the option **direct_camp_on = never** was selected. When creating multiple rotaries under one name, always disable camp-on.

Since a modem was not available on the first Annex, **telnet** automatically crossed over to the second Annex, *annex05*.

Assigning Rotaries TCP Port Numbers

TCP port numbers in the 6000 range allow you to assign a TCP port number to a rotary that the user can enter with the **telnet** command. The last three digits of the port number are arbitrary; but the TCP port numbers must be unique for each Annex.

When users include a TCP port number in the 6000 range with the **telnet** command, they are attached to the first available port in the rotary. Defining TCP ports for rotaries allows the users to avoid having to select a particular serial port, especially if auxiliary Internet addresses cannot be used.

The following example is an entry in which a TCP port number is defined for the rotary:

```
modems: 1,4,6-9@annex01/6080
```

The user can issue the following command to access the first available port in the rotary. For example:

```
% telnet annex01 6080
Trying...
    Connected to annex01
    Escape character is '^]'.
    Attached to port 9
```

Configuring Visibility

A visible rotary is a rotary whose name is displayed by the port server when users **telnet** to the Annex's primary Internet address. An invisible rotary prevents users from seeing the name if they **telnet** to the Annex and further hides the details of the connection.

Only rotaries that can be accessed via an auxiliary Internet address or a TCP port in the 6000 range can be defined as invisible. Rotaries without auxiliary Internet addresses or TCP ports in the 6000 range are always visible.

Following is an example of an entry that makes the *HostC* rotary invisible:

```
HostC: ps_invisible 1-4,29-32@annex03+132.245.6.33
```

Users that **telnet** to *annex03* do not see the name; users that **telnet** to *HostC* see the sequence illustrated in *Assigning Rotaries Internet Addresses*.

Configuring Camp-on

Camp-on can be configured for rotaries that are accessed through an auxiliary Internet address or TCP port. The options are: **ask, always,** and **never.** The default is **ask** (except for raw rotaries).

The option **ask** causes the port server to query the user whether or not to camp-on; **never** causes the port server to refuse the connection if all ports are busy; **always** causes the port server to camp-on automatically.

Configuring Raw Rotaries

A raw rotary performs no data processing; data is passed directly to and from the serial device. Generally, raw rotaries are accessed by programs that use the system network facilities, such as the **socket** interface in BSD systems, to open a connection and to perform whatever functions are required for the device.

Raw rotaries are defined with the keyword **raw.** Raw rotaries are always invisible. Camp-on can be configured using either the **never** or the **always** option. The **never** option is the default for raw rotaries; **ask** cannot be used with raw rotaries.

Following is an example of a raw rotary consisting of ports 1, 2, 3, and 8 on an Annex whose Internet address is 132.245.6.32. The rotary is accessed through TCP port 6300:

```
strip-record: raw direct_camp_on=always 1-3,8@132.245.6.32/6300
```

Configuring Binary Rotaries

If you want a rotary to operate in **telnet binary** mode, set the keyword in the rotaries file to **binary.** In this configuration, the Annex negotiates with the host to operate in **telnet binary** mode in both directions.

Compiling the Rotaries File

The Annex downloads from the load host rotary information from a file called **rotaries.** The Annex extracts its assigned rotary definitions from that file.

Note: The **rotaries** file must be installed in the directory **/usr/spool/erpcd/bfs.**

The Annex expects the preferred load host to provide the **rotaries** file. If this host does not respond with the file, the Annex broadcasts for it (unless **load_broadcast** is disabled or set to **N**). If the file is not found when an Annex boots, no rotaries are defined on that Annex.

Chapter 4

Printers

General

The Annex provides the following mechanisms for hosts to send print requests to a remote printer:

- **aprint**: a simple printing utility used to send print jobs to the parallel printer port or unidirectional printers. Printers using **aprint** must be output only devices.
- **rtelnet**: the reverse **telnet** daemon used for bidirectional printers, such as PostScript printers, for printing software that expects to talk to a **/dev** device, or for printing packages that are expecting a tty device.

Although the **aprint** and **rtlenet** utilities do not provide spooling capabilities, you can use them as transparent utilities for other printer spooling systems (e.g., Berkeley UNIX **lpd** or System V **lp**).

This chapter describes:

- The **na** parameters for configuring a printer port
- Host configuration for BSD hosts
- Host configuration for System V hosts

Configuring Ports for Printers

You can attach a printer to the Annex's parallel port, which supports either a Centronics- or Dataproducts-compatible printer, and to the serial ports. This section describes the port parameter settings for both types of printers.

Configuring Port Parameters for a Serial Printer

Configure the printer for use with a serial port. Chapter 2 describes setting port parameters using **na**. Consider the following **na** parameters when attaching a printer to a serial port:

- Set the **type** parameter to **hardwired.**
- Set the **mode** parameter to **slave.**
- Set the **speed** parameter to the speed of the printer. Do not use **autobaud.**
- Set the **data_bits, stop_bits,** and **parity** parameters to match the requirements of the printer.
- If the printer supports EIA flow control, set the **control_lines** parameter to **flow_control** and the **input_flow_control** and **output_flow_control** parameters to **eia.** If a printer supports XON/XOFF (cannot be wired for EIA) set the **control_lines** parameter to **none** and the **output_flow_control** parameter to **start/stop.**
- Set the **allow_broadcast** parameter to N to disable the display of administrative messages on this port.

Configuring Printer Parameters for a Parallel Printer

To configure the parallel printer port, use the **set printer** command. The parallel port parameters are **hardware_tabs, map_to_upper, printer_width,** and **type.**

If the printer is using a Centronics interface, use the supplied default, **centronics,** for the **type** parameter. If the printer is using a Dataproducts printer interface, set this parameter to **dataproducts** (the Dataproducts interface is supported only on the Annex IIe and *Annex 3.*)

Note: This **type** parameter is not the same parameter as the serial port **type** parameter.

The **show printer** command displays the valid parallel printer port parameters (the **map_to_upper** parameter has no effect).

BSD Host Configuration

The Annex software provides three options for configuring a BSD host to send print requests to a remote printer attached to an Annex: 1) the **aprint** utility, 2) the **rtelnet** utility, and 3) modifying the Berkeley print spooler.

Setting Up aprint on BSD Hosts

The **aprint** utility sends files directly to an Annex printer connected to the parallel port or to a serial port. The **aprint** utility can be used in one of two ways: 1) as part of an output filter of a BSD spooling system, and 2) as a direct command.

The Filter Program filt.c

The *C* program **filt.c** was designed for use as a *filter* in the **/etc/printcap** file. The program sends data to the Annex port specified in the program name. The program name should conform to: **annexname.port**.

Note: The **filt.c** program is compatible with SunOS; other UNIX operating systems may require modifications to the program.

The **filt.c** program looks like this:

```
#include <stdio.h>
#define SEPARATOR       '.'
#define APRINT          "/usr/annex/aprint"

/*
 * If you define a filter end string with |
 */

#define    FILTER       ""
main(argc,argv)
int argc ;
char *argv[] ;
{
    char                annex[20];
    char                port[20];
    char                line[120];
    char                *p;
    int                 length;

    p = (char *)strrchr(argv[0],SEPARATOR);
    length = (int)(p-argv[0]);
    strncpy(annex,argv[0],length);
    annex[length] = '\0';
    strcpy(port,p+1);

    sprintf(line,"%s %s -f -A%s -L%s",FILTER,APRINT,annex,port) ;
    system(line);

    /* Always return OK to the spooler daemon */
    exit(0) ;
}
```

Using aprint with an Output Filter

The **aprint** program is not a filter; it cannot perform any conversions or expansions. However, you can set up **aprint** as part of an output filter for the BSD **lpd** program and include other filters for conversion and expansion. The filters to be used are defined by the FILTER definition at the top of the program. The example given does no mapping.

For data that needs tab expansion and newline to carriage return/line feed conversion, change the definition of FILTER to:

```
#define FILTER "awk'{printf(\"%s\\r\\n\",$0) }' | expand | "
```

Create this output filter using the *C* program **filt.c** that calls the real filter program(s), and pipes the output to **aprint**. Compile and link the **filt.c** program to a name that specifies the Annex port to which the output should go. To use **aprint** in this way, use the **of=** argument in the **/etc/printcap** file to specify the final output filter. For example, to send output to port 12 on the Annex called *annex01*, **filt.c** should be linked to *annex01.15*. If the *annex01.15* program is in the directory **/usr/annex**, the entry in **/etc/printcap** would look like this:

```
annexprt|ap|Annex printer\
    :lp=/dev/null:sd=/usr/spool/annexprt:\
    :lf=/usr/adm/lpd-errs:\
    :of=/usr/annex/annex01.15
```

If you are using more than one printer, some versions of BSD prevent more than one printer from running at once because **/dev/null** is used as the device name for all printers. In this case, each printer must use a unique name; create a unique copy of **/dev/null** for each printer in use using the **mknod** program:

```
mknod /dev/null0s c 3 2
```

where *3 2* gives the major and minor device numbers of **/dev/null**. Find these numbers within your system because they differ from manufacturer to manufacturer. Use the new name instead of **/dev/null** in the previous example.

Using aprint as a Direct Command

When using **aprint** as a direct command, the user specifies the Annex and port with the **-A** and **-L** arguments. If the printer is attached to the parallel port, do not include the **-L** argument. With **aprint** a banner page is not printed. If you want a banner page, create a simple shell script that calls **aprint**. If you place the shell in a public executable area, users can execute the shell instead of calling **aprint**. The script must take the file(s) to be printed as an argument, add a banner page, and call **aprint** using the **-A** and **-L** arguments. The following is an example of a script called **laser1**:

```
#!/bin/sh
(
    makebanner $*
    cat $*
) | aprint -Aannex01 -L12
```

Users can then issue the command script **laser1** to print. For example:

```
# laser1 file1 file2
```

Note: The utility *makebanner* is not supplied; it is an example.

Setting Up rtelnet on BSD Hosts

The **rtelnet** daemon provides a Telnet connection between an Annex port and a character device on a host. This connection provides bidirectional communications for accessing printers on a serial port. You can set up **rtelnet** to start when the host boots after which **rtelnet** provides a link from a **/dev** file on the host to an Annex port. To set up an **rtelnet** daemon to be used by the BSD LPD spooling system:

1. Create a special file with **rtelnet**. For example:

   ```
   rtelnet -br annex02 16 /dev/annexprt
   ```

   In this example, the **-b** argument is included because the printer uses binary data which may be scrambled by Telnet's CR/LF conventions.

2. Add an entry to the **/etc/printcap** file for the printer. This entry should appear as if the printer is connected directly to a host serial port called **/dev/annexprt**. For example:

   ```
   annexprt|ap|serial printer on annex02:\
       :lp=/dev/annexprt:\
       :sd=/usr/spool/annexprt:\
       :lf=/usr/adm/lpd-errs:pw#80:fs
   ```

3. Test the printer:

   ```
   # lpr -Pannexprt /etc/printcap
   ```

System V Host Configuration

The Annex software provides two utilities for configuring a System V host for sending print jobs to a printer attached to an Annex: **aprint** and **rtelnet.**

For either utility, use **/usr/lib/lpadmin** to configure the System V **lp** spooling system for printers attached to Annexes. Annex printers use the following format:

lpadmin -p*printer* **-v***device* [**-e***printer* | **-i***interface* | **-m***model*]

The **-e** option copies the interface program of the specified printer to the new printer specified with **-p**. The **-i** option defines the full pathname for a new interface program. The **-m** option specifies a model interface program for the printer. The specified *model* is one of the model interfaces supplied with the System V **lp** spooling utilities.

Setting Up aprint on System V Hosts

The **aprint** utility is used for printers attached to the parallel printer port on the Annex or for unidirectional serial line printers. You can use the **aprint** utility either with an interface file or as a direct command.

Using aprint with an Interface File

When an interface file is used with **aprint**, that file is almost identical to the standard interface file for a System V printer except that all output destined for the printer is piped into a single instance of **aprint.**

The following set of instructions specifies the steps necessary to set up each type of printer. The following example uses an interface file with the path name **/usr/spool/lp/annex_printer** that was specifically created for a particular printer called *p_annex*. An example of this file is provided at the end of the chapter.

1. Shut down the **lp** scheduler:

```
/usr/lib/lpshut
```

2. Define a new printer using the **/usr/lib/lpadmin** command. For example:

```
lpadmin -pp_annex -v/dev/null -i/usr/spool/lp/annex_printer
```

3. Enable the printer. For example:

```
enable p_annex
```

4. Allow the queue to accept jobs. For example:

```
/usr/lib/accept p_annex
```

5. Restart the **lp** scheduler.

Using aprint as a Direct Command

When using **aprint** as a direct command, the user specifies the Annex and port with the **-A** and **-L** arguments. (If the printer is attached to the parallel port, do not include the **-L** argument.) The **aprint** utility does not print a banner page. If you want a banner page, create a shell script that calls **aprint**. If you place the shell in a public executable area, users can use the shell instead of calling **aprint**.

The script must take the file(s) to be printed as an argument, add a banner page, and call **aprint** using the **-A** and **-L** arguments. For example, a script called **laser1**:

```
#!/bin/sh
(
    makebanner $*
    cat $*
) | aprint -Aannex01 -L12
```

Users can then issue the command script **laser1** to print. For example:

```
laser1 file1 file2
```

Note: The utility *makebanner* is not supplied; it is an example.

Setting Up rtelnet on System V Hosts

A host can access a printer attached to an Annex through a character special file created using the **rtelnet** utility. The following example is for a DQP-10 printer using the **dqp10** model interface; in this example, the printer is called *s_annex*.

Note: The **rtelnet** utility does not run on all versions of System V.

1. Create a printer interface script for the type of printer.
2. Shut down the **lp** scheduler:

```
/usr/lib/lpshut
```

3. Create a character special device with **rtelnet**. For example:

```
rtelnet -br annex01 12 /dev/s_printer
```

   This command creates the character special file **/dev/s_printer** and creates a Telnet connection between the associated pty device and port 12 on the Annex called *annex01*.

4. Define a new printer using the **lpadmin** command. For example:

```
lpadmin -ps_annex -v/dev/s_printer -mdqp10
```

5. Enable the printer. For example:

```
enable s_annex
```

6. Allow the queue to accept jobs. For example:

```
/usr/lib/accept s_annex
```

7. Restart the **lp** scheduler.

Sample System V Interface File for aprint

The following example of an interface file illustrates setting up **aprint** on a System V host to pipe the output destined for the printer through a single instance of **aprint**.

Note: If you require tab expansion and newline carriage return/line feed conversion, change the line:

```
aprint -A$ANNEX -L$ANNEXLINE -f
```

to:

```
awk'{ printf("%s\r\n",$0) }' |
expand |
aprint -A$annex -L$ANNEXLINE -f
```

```
#
# lp interface for line printers
#
# SCCS @(#) lp 1.2

ANNEX=annex01
ANNEXLINE=0
# 0 for parallel printer, port number for serial
# Change this line if your annex software is
# installed in a different directory
PATH=$PATH:/usr/annex
export PATH

# This will be executed when a request is cancelled

trap "echo '\n\n\n\nRequest Cancelled'; \
      echo '\014\c'; \
      sleep 30; \
      exit 0" 15

#
# THE OPEN PARENTHESIS APPLIES ONLY TO APRINT
#
(
#
x="XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX"
echo "\014\c"
echo "$x$x\n$x$x\n$x$x\n"
banner "$2"
echo "\n"
user=`grep "^$2:" /etc/passwd | line | cut -d: -f5`
if [ -n "$user" ]
then
    echo "User: $user\n"
else
    echo "\n"

fi
echo "Request id: $1 Printer: `basename $0`\n"
date
echo "\n"
if [ -n "$3" ]
then
    banner $3
fi
copies=$4
echo "\014\c"
shift;shift;shift;shift;shift
files="$*"
i=1
while [ $i -le $copies ]
do
    for file in $files
    do
            cat "$file" 2>&1
            echo "\014\c"
    done
    i=`expr $i + 1`
done

#
# THE FOLLOWING TWO LINES APPLY ONLY TO APRINT
#
) |
  aprint -A$ANNEX -L$ANNEXLINE -f

exit 0
```

Chapter 5

Modems

General

This chapter describes how to configure port parameters for modems and for devices that behave like modems (e.g., serial line switches).

Modem Configurations

A modem on an Annex can be configured in one of three ways:

- Outbound: modems that initiate only outgoing calls.
- Inbound: modems that accept only incoming calls.
- Bidirectional: modems that accept and initiate calls.

Modem Signals

Because each model of the Annex has a different hardware platform, support for modem signals differs among them. The following subsections describe the modem signals for the Annex 3, the Annex II, the Annex IIe, and original Annexes.

Annex 3

The Annex 3 has three input and two output signals. The use of these signals is determined by the **control_lines, input_flow_control, output_flow_control bidirectional_modem**, and **need_dsr** port parameters.

To use modem control (DTR/DCD/DSR), set the **control_lines** parameter to **modem_control**. The Annex 3 asserts DTR when the port is ready for use by the Annex. It then waits for DCD and DSR to be asserted before opening the session. After opening the session, any drop of DCD that lasts more than 400 milliseconds, or any drop of DSR, causes a port reset. A port reset drops DTR for at least one second and kills any jobs attached to the port.

To use EIA/hardware flow control (RTS/CTS), set the **control_lines** parameter to **flow_control**, and the **input_flow_control** and **output_flow_control** parameters to **eia**. The Annex 3 asserts RTS when it is ready to receive data, and checks the CTS input before transmitting data.

To use software flow control (XON/XOFF), set **control_lines** to **none**, and set both **input_flow_control** and **output_flow_control** to **start/stop**. The Annex 3 sends XON when it is ready to receive data, and XOFF to suspend transmission

To use both EIA/hardware flow control (RTS/CTS) and modem control (DTR/DCD/DSR), set **control_lines** to **both**, and **input_flow_control** and **output_flow_control** to **eia**. The Annex 3 uses these signals as described in the previous paragraphs.

To use both software flow control (XON/XOFF) and modem control (DTR/DCD/DSR), set **control_lines** to **modem_control**, and **input_flow_control** and **output_flow_control** to **start/stop**.

The **bidirectional_modem** parameter configures the port for adaptive use with a bidirectional modem. When enabled (set to **Y**), the Annex allows connections to the port without waiting for DCD.

When using a modem connected to a slave port, if the **need_dsr** parameter is enabled, the connection fails if no DSR signal is present; if **need_dsr** is disabled, the Annex accepts the connection, but waits for DSR and DCD before communicating with the modem; if **bidirectional_modem** is enabled, it only waits for DSR.

Annex IIe

The Annex IIe has two input and two output signals. The use of these signals is determined by the **control_lines**, **input_flow_control**, **output_flow_control** and **bidirectional_modem** port parameters.

Note: The Annex IIe can be configured via jumpers to use the same signals as an Annex II. See your hardware guide for more details.

To use modem control (DTR/DCD), set **control_lines** to **modem_control**. The Annex IIe asserts DTR when the port is ready for use by the Annex. It then waits for DCD and DSR to be asserted before opening the session. After opening the session, any drop of DCD that lasts more than 400 milliseconds causes a port reset. A port reset drops DTR for at least one second and kills any jobs attached to the port.

To use EIA/hardware flow control (RTS/CTS), set the **control_lines** parameter to **flow_control**, and the **input_flow_control** and **output_flow_control** parameters to **eia**. The Annex IIe asserts RTS when it is ready to receive data, and checks the CTS input before transmitting data.

To use software flow control (XON/XOFF), set **control_lines** to **none**, and set both **input_flow_control** and **output_flow_control** to **start/stop**. The Annex IIe sends XON when it is ready to receive data, and XOFF to suspend transmission.

To use both EIA/hardware flow control (RTS/CTS) and modem control (DTR/DCD), set **control_lines** to **both**, and **input_flow_control** and **output_flow_control** to **eia**. The Annex IIe uses these signals as described in the previous paragraphs.

To use both software flow control (XON/XOFF) and modem control (DTR/DCD), set **control_lines** to **modem_control**, and **input_flow_control** and **output_flow_control** to **start/stop**.

The **bidirectional_modem** parameter configures the port for adaptive use with a bidirectional modem. When enabled (set to **Y**), the Annex allows connections to the port without waiting for DCD.

Original Annex and Annex II

The original Annex and the Annex II have one input and one output signal. The use of these signals is determined by the **control_lines** and **bidirectional_modem** port parameters.

To use modem control (DTR/DCD), set **control_lines** to **modem_control**. Both the original Annex and the Annex II assert DTR when the port is ready for use by the Annex. It then waits for DCD to be asserted before opening the session. After opening the session, any drop of DCD that lasts more than 400 milliseconds causes a port reset. A port reset drops DTR for at least one second and kills any jobs attached to the port.

To use EIA/hardware flow control (RTS/CTS), set the **control_lines** parameter to **flow_control**, and the **input_flow_control** and **output_flow_control** parameters to **eia**. Both the original Annex and the Annex II assert RTS when they are ready to receive data, and check the CTS input before transmitting data.

The **bidirectional_modem** parameter configures the port for adaptive use with a bidirectional modem. When enabled (set to **Y**), the Annex allows connections to the port without waiting for DCD.

Port Parameters for Modems

The following subsections discuss parameters that, if not set, can cause problems and parameters used to access Annex features.

Setting Port Parameters for Outbound Modems

- Set the **mode** parameter to **slave.**
- Set the **type** parameter to **dial_in.**
- Set the **control_lines, input_flow_control,** and **output_flow_control** parameters as desired. See *Modem Signals* at the beginning of this chapter for details.
- The default setting for software flow control is **start/stop** (XON/XOFF). This setting may not be suitable for file transfer applications, such as **uucp, kermit,** and **xmodem.**
- The **bidirectional_modem** parameter configures the port for adaptive use with a bidirectional modem. When enabled (set to **Y**), the Annex allows connections to the port without waiting for DCD.
- When using a modem connected to a slave port, if the **need_dsr** parameter is enabled, the connection fails if no DSR signal is present; if **need_dsr** is disabled, the Annex accepts the connection, but waits for DSR and DCD before communicating with the modem; if **bidirectional_modem** is enabled, it only waits for DSR.
- Set the **speed** parameter to the speed of the modem. Do not use **autobaud.**
- Set the **data_bits, stop_bits,** and **parity** parameters to match the requirements of the modem.
- Set the **broadcast_direction** parameter to **network.** This sends all administrative broadcasts to users connected to the modem port.
- The **cli_inactivity** parameter is a timer that resets the port after all CLI sessions have exited.
- Set the **inactivity_timer** to limit the amount of time the modem can be left idle with a connection. When this amount of time passes with no activity detected, the Annex drops DTR, terminates the port server connection, and resets the port. The inactivity timer hangs up the port, releasing it for another outgoing call. The default setting for this parameter is **off.** Set the **output_is_activity** parameter to **Y.** With an outbound modem, activity should be defined as output.
- Enable security on the port using the **port_server_security** parameter and/or the **port_password** parameter to prevent unauthorized access through the port server. If **rtelnet** is used to access the port, the host application, **tip, cu, uucp,** etc., must be configured to pass the password to the security subsystem.

The following example shows how to configure port 13 for a 2400 bps outbound modem:

```
command: port 13
command: set port speed 2400 control_lines modem_control\
         type dial_in mode slave inactivity_timer 15\
         output_is_activity Y broadcast_direction network\
         input_flow_control start/stop\
         output_flow_control start/stop
```

Setting Port Parameters for Inbound Modems

- Set the **type** parameter to **dial_in**.
- Set the **control_lines, input_flow_control,** and **output_flow_control** parameters as desired. See *Modem Signals* at the beginning of this chapter for details.
- The default setting for software flow control is **start/stop** (XON/XOFF). This setting may not be suitable for file transfer applications, such as **uucp, kermit,** and **xmodem** .
- The **bidirectional_modem** parameter configures the port for adaptive use with a bidirectional modem. When enabled (set to **Y**), the Annex allows connections to the port without waiting for DCD.
- When using a modem connected to a slave port, if the **need_dsr** parameter is enabled, the connection fails if no DSR signal is present; if **need_dsr** is disabled, the Annex accepts the connection, but waits for DSR and DCD before communicating with the modem; if **bidirectional_modem** is enabled, it only waits for DSR.
- Set the **mode** to **dedicated** and define the host to which this connect is made using the **dedicated_address** parameter. The default setting for the **dedicated_port** parameter is **telnet.**
- Set the **speed** parameter to the speed of the modem. If you set the speed to **autobaud,** the user must press the **Return** key several times for the Annex to determine the speed.
- Set the **data_bits, stop_bits,** and **parity** parameters to match the requirements of the modem.
- Set the **cli_imask7** parameter to **Y** if a device dialing in sends seven bits with parity under normal interactive use, but sends eight bits when transferring binary files (some hosts, especially PCs do this). When **cli_imask7** is set to **Y,** input to the CLI is masked to seven bits. When no longer at the CLI, input is not masked. If you are expecting eight-bit ASCII input, set **cli_imask7** to **N.**
- By setting the **mode** to **cli,** many of the remaining port parameters can be configured for CLI-connected terminals (see *Chapter 2: Configuring Ports* in this book for more information regarding setting port parameters using na).
- If the port **mode** is set to either **cli** or **dedicated,** the typical setting for the **term_var** parameter is **dialup.**
- The **cli_inactivity** parameter is a timer that resets the port after all CLI sessions have exited.

- Set the **inactivity_timer** to limit the amount of time the modem can be left idle with a connection. When this amount of time passes with no activity detected, the Annex drops DTR, terminates the port server connection, and resets the port. The inactivity timer hangs up the port, releasing it for another outgoing call. The default setting for this parameter is **off**. Set the **input_is_activity** parameter to **Y**.
- Define a control character for the **attn_char** parameter because modems do not always handle the **Break** key appropriately (i.e., signal the Annex to suspend the session with the host and return to the CLI prompt). However, the **attn_char** can interfere with file transfer programs (**uucp, kermit, xmodem**). Set the parameter to off or the remote user must issue the CLI **stty** command to turn off the parameter.
- If the port **mode** is set to **cli**, enable CLI and/or connect security using the port security parameters:**cli_security, connect_security**, and **port_password**.

The following example shows how to configure a port for a 2400 bps inbound modem as a CLI port:

```
command: port 12
command: set port speed 2400 control_lines modem_control\
         type dial_in mode cli inactivity_timer 15\
         term_var dialup user modem cli_security Y\
         cli_inactivity 10
```

An example of using **na** to configure a port for a 2400 bps inbound modem as a slave port is:

```
command: port 16
command: set port speed 2400 control_lines modem_control\
         type dial_in mode slave inactivity_timer 15\
         output_is_activity Y
```

Setting Port Parameters for Bidirectional Modems

- Set the **mode** parameter to **adaptive**.
- Set the **type** parameter to **dial_in**.
- Set the **control_lines, input_flow_control**, and **output_flow_control** parameters as desired. See *Modem Signals* at the beginning of this chapter for details.
- The default setting for software flow control is **start/stop** (XON/XOFF). This setting may not be suitable for file transfer applications, such as **uucp, kermit**, and **xmodem**. Optionally, remote users can issue the CLI **stty** to turn off flow control.

- The **bidirectional_modem** parameter configures the port for adaptive use with a bidirectional modem. When enabled (set to **Y**), the Annex allows connections to the port without waiting for DCD.
- When using a modem connected to a slave port, if the **need_dsr** parameter is enabled, the connection fails if no DSR signal is present; if **need_dsr** is disabled, the Annex accepts the connection, but waits for DSR and DCD before communicating with the modem; if **bidirectional_modem** is enabled, it only waits for DSR.
- Set the **speed** parameter to the speed of the modem. Do not use **autobaud**.
- Set the **data_bits**, **stop_bits**, and **parity** parameters to match the modem's requirements.
- The typical setting for the **term_var** parameter is **dialup**.
- The **cli_inactivity** parameter is a timer that resets the port after all CLI sessions have exited.
- Set the **inactivity_timer** to limit the amount of time the modem can be left idle with a connection. When this amount of time passes with no activity detected, the Annex drops DTR, terminates the port server connection, and resets the port. The inactivity timer hangs up the port, releasing it for another outgoing call. The default setting for this parameter is **off**. Set the **output_is_activity** parameter to **Y**. With an outbound modem, activity should be defined as output.
- Set both the **input_is_activity** and **output_is_activity** parameters to **Y**.
- If the port **mode** is set to **cli**, enable CLI and/or connect security using the port security parameters:**cli_security**, **connect_security**, and **port_password**.
- Define a control character for the **attn_char** parameter because modems do not always handle the **Break** key appropriately (i.e., signal the Annex to suspend the session with the host and return to the CLI prompt). However, the **attn_char** can interfere with file transfer programs (**uucp**, **kermit**, **xmodem**). Set the parameter to off or the remote user must issue the CLI **stty** command to turn off the parameter.

An example of configuring an Annex 3 port for a 9600-baud bidirectional modem is:

```
command: port 16
command: set port speed 9600 control_lines both\
         type dial_in mode adaptive bidirectional_modem Y\
         output_is_activity Y inactivity_timer 20\
         input_flow_control eia output_flow_control eia\
         cli_security Y location V24BIS
```

Host Set Up Procedures

The following subsections provide procedures for setting up host files that allow applications to access the Annex port on which a modem is attached. These procedures use the **rtelnet** utility (see *Book D: Reference, Chapter 4: Utilities*).

Outbound Modems

Set up files on a host if you want applications, such as **tip** and **uucp**, to access the outbound modem. To set up these files:

1. Use the Annex **rtelnet** utility to create a special file that references the port. For example:

   ```
   # rtelnet -mr annex02 13 /dev/modem
   ```

   In this example, the **-m** argument is required and instructs the Annex to drop momentarily the network connection to the Annex port when the pseudo device is closed. This argument causes the modem to hang up when the application exits.

   The **-r** argument is recommended and directs **rtelnet** to remove the device *device name* if it already exists. Without **-r**, **rtelnet** exits with an error message if *device name* already exists.

2. Add the **rtelnet** command to **/etc/rc** so that the special file is created when the host boots.

3. Configure the application to use the special file created by **rtelnet** to access the modem. (See the appropriate system documentation for the required steps.)

   Using **rtelnet** allows the application to be set up as if **/dev/modem** is a directly connected device.

Note: If the system uses a name server to translate host names to Internet addresses and you elect to use the Annex's name in the **rtelnet** command, make sure that the Annex is listed in the name server database and that the name server is started before the **rtelnet** command.

Inbound Modems

Set up files on a host if you want applications, such as **getty**, to access the inbound modem. To set up these files:

1. Use the Annex **rtelnet** utility to create a special file that references the port. For example:

   ```
   # rtelnet annex02 3 /dev/ttyDB
   ```

2. Enable a **getty** process that monitors the port. See the appropriate system documentation for the required steps.

   For a 4.3BSD system, add a line to **/etc/ttys** to define the *device name* as a **getty** line. For a Sun OS 4.*x* system, add a line to **/etc/ttytab**. The following example can be used with either system:

   ```
   ttyDB "/etc/getty std.9600" ansi on
   ```

   Then, signal **init** so it reads the file and starts a **getty** now:

   ```
   # kill -HUP 1
   ```

Note: Other operating systems use different formats for creating a **getty** process.

3. Add the **rtelnet** command to **/etc/rc** so that the special file will be created when the system is booted.

Note: If the system uses a name server to translate host names to Internet addresses and you elect to use the Annex's name, make sure that the Annex is listed in the name server database and that the name server is started before the **rtelnet** command.

Bidirectional Modems

Set up files on a host if you want applications, such as **tip** and **uucp**, to access the modem. To set up these files:

1. Use the Annex **rtelnet** utility to create a special file that references the port. For example:

   ```
   # rtelnet -fmr annex02 13 /dev/modem
   ```

   In this example, the **-f** argument is required and instructs the Annex to release the port when the device is no longer using it, thus releasing the port for use as a CLI.

The **-m** argument is required and instructs the Annex to drop momentarily the network connection to the Annex port when the pseudo device is closed. This argument causes the modem to hang up when the application exits.

The **-r** argument is recommended and directs **rtelnet** to remove the device *device name* if it already exists. Without **-r**, **rtelnet** exits with an error message if *device name* already exists.

2. Add the **rtelnet** command to **/etc/rc** so that the special file is created when the host boots.
3. Configure the application to use the special file created by **rtelnet** to access the modem. See the appropriate system documentation for the required steps.

   Using **rtelnet** allows the application to be set up as if **/dev/modem** is a directly connected device.

Note: If the system uses a name server to translate host names to Internet addresses and you elect to use the Annex's name in the **rtelnet** command, make sure that the Annex is listed in the name server database and that the name server is started before the **rtelnet** command.

Chapter 6

Serial Line Internet Protocol (SLIP)

General

The Serial Line Internet Protocol (SLIP) enables you to send TCP/IP packets across a serial line. A SLIP link is a point-to-point connection between two hosts. Using a SLIP link, you can multiplex expensive leased-line connections between distant networks, or connect a host to the network without requiring special interface hardware. The Annex implementation of SLIP is compatible with the 4.3BSD implementation. The Annex also supports Compressed SLIP as an option.

The Annex provides several options for setting up a SLIP link:

- You can use SLIP to connect two separate networks, routing data from one network to the other over the SLIP link. SLIP can also be used to connect remote Annexes that support terminals, printers, etc., or that support remote hosts over serial lines.
- You can attach a PC to an Annex serial port. Using SLIP as the network interface, the PC becomes a host on the network. For a remote PC, you can configure a port as both a SLIP link and an incoming modem. Then, a user at a remote PC can dial into the Annex and convert the port from an incoming modem to a SLIP link using the CLI **slip** command. After converting the port to a SLIP link, the remote PC becomes a host directly attached to the network.
- You can use a SLIP link to boot and, optionally, dump an Annex. You can define a list of SLIP links and the local network interface over which a download is to be done. During a load, the Annex makes a request from the first interface in the list. If the host associated with that interface does not respond, the Annex makes the request from next interface in the list, and so forth until it has been booted.

  Booting over a SLIP link supports remote Annexes, and allows a PC to act as a load host for the Annexes on the network. Any serial port can be configured for SLIP.

Compressed SLIP

The Compressed Serial Line Internet Protocol (CSLIP) improves bandwidth by compressing the TCP/IP headers from 40 bytes to as few as three bytes when running over a SLIP link. Compression creates smaller packets, and therefore faster throughput.

You can choose either a configuration that uses compressed SLIP always, or one that uses compressed SLIP only when the remote end sends compressed SLIP packets. The Annex's implementation of CSLIP offers four options:

- Do Compressed SLIP.
- Allow Compressed SLIP.
- Discard ICMP requests over the SLIP link.
- Give interactive traffic priority over other traffic.

SLIP Configurations

When using SLIP to connect two networks together or to connect a single host to the network, you must assign Internet addresses to both ends of the SLIP link.

Connecting Two Networks Together

When connecting two networks together, you can:

- Assign a separate Internet network or subnet address to the SLIP link.
- Use the Internet address that is assigned to the hosts at each end of the SLIP link.

The following figures illustrate Class B subnetted networks. Figure B–3 illustrates a network with a separate subnet address. In this example, the SLIP link is assigned a separate network address of 132.245.99 and is treated like any other physical network. In this configuration you lose a network number.

If you are conserving network numbers, you may prefer the configuration depicted in Figure B–4. Using this option, the Internet addresses assigned to the end-points of the SLIP link are the hosts' primary network Internet addresses.

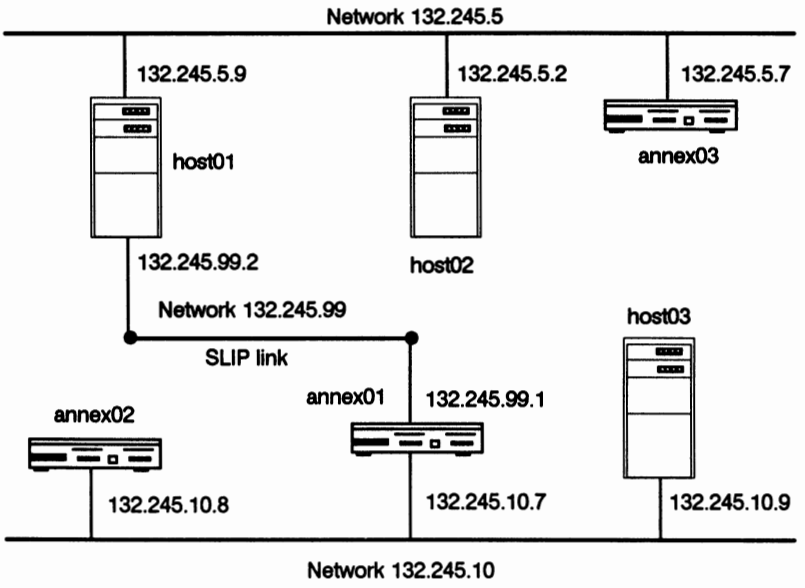

Figure B–3. SLIP Link with Separate Network Address

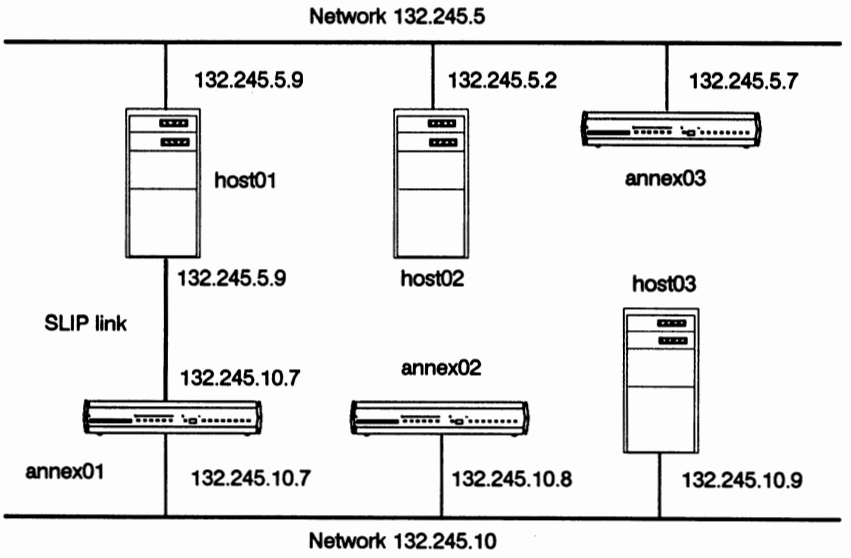

Figure B–4. SLIP Link with Two Internet Addresses

Connecting a Single Host with a SLIP Link

In Figure B–5, a single host connected to an Annex through a SLIP link appears to the network as an attached host. Assign that host a unique host address for the network address.

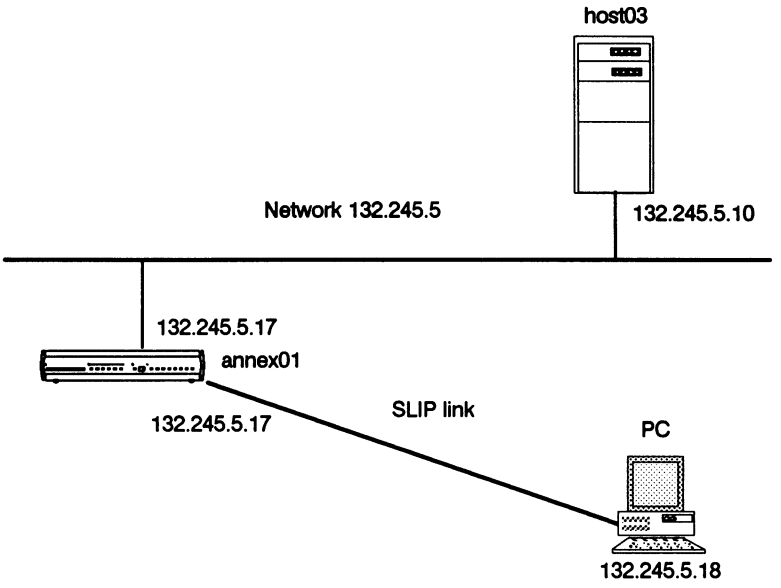

Figure B–5. Connecting a Single Host Through SLIP

Connecting a Remote Annex

In Figure B–6, a remote host connected to an Annex through a SLIP link appears as a local Annex to the network. Assign the remote Annex a unique host address for the network address.

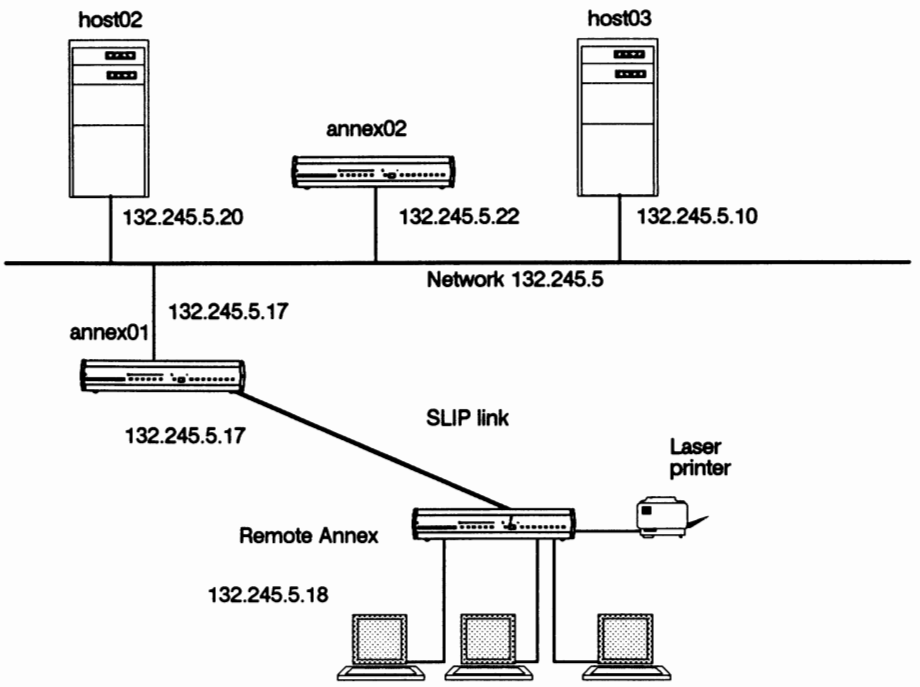

Figure B–6. Connecting a Remote Annex

Setting Ports for a SLIP Interface

This section describes how to set port parameters for different SLIP configurations.

Note: Original Annexes do not support SLIP port parameters.

SLIP Configurations for a Host

- Use the supplied defaults for the **data_bits** (8), **stop_bits** (1), and **parity** (none) parameters.
- Set the **speed** parameter to the rate required for the SLIP link. Set the remote end to match these values.

- Set the **mode** parameter to **slip**. Set the **type** parameter to **hardwired.**
- If the device at the remote end of the SLIP interface supports EIA flow control, you can configure the port for EIA flow control. Set the **control_lines** parameter to **flow_control** and the **input_flow_control** and the **output_flow_control** parameters to **eia.** Do not set these parameter to **start/stop.**

Set the appropriate SLIP parameters. These parameters are set with the **set port** command, and are displayed with the **show port slip** command.

- The **slip_local_address** parameter defines the Internet address for this port. This address is the address for the Annex's side of the SLIP link. For example, in Figure B–4 the **slip_local_address** for *annex01* is 132.245.10.7, which is the same address specified with the Annex's **inet_addr** parameter. In Figure B–3, the **slip_local_address** for *annex01* is 132.245.99.1.
- The **slip_remote_address** parameter defines the Internet address for the host's end of the SLIP link.
- The **slip_subnet_mask** parameter defines the subnet mask if the SLIP link is using a subnet.
- The **slip_metric** parameter defines the cost, or how many hops to get to the remote end of the SLIP link. You may want to increase this number if the Annex has an alternate, higher-bandwidth route to the remote host.
- The **slip_do_compression** parameter forces TCP header compression on the SLIP link. When enabled, the Annex always sends compressed packets and the other end of the SLIP link also must send compressed packets.
- The **slip_allow_compression** parameter allows the Annex compress TCP headers if the other end of the SLIP link also provides compression. When enabled, and the other end of the SLIP link sends compressed packets, the Annex sends compressed packets. If the other end of the SLIP link cannot compress packets, the Annex sends normal SLIP packets.
- The **slip_no_icmp** parameter discards any ICMP pakets that are destined for the SLIP link, thereby reducing unnecessary traffic and messages over the SLIP link.
- The **slip_tos** parameter causes the Annex to send interactive traffic (**telnet, rlogin,** and ftp control sessions) before any other traffic over the SLIP link; it provides a type-of-service queuing on the link.

To use the SLIP link for booting and/or dumping, you need to set an Annex parameter as well as the following SLIP parameters.

- The **slip_load_dump_host** parameter defines the Internet address from which the Annex requests boots and to which the Annex dumps. This parameter can be the same Internet address that you specified in the **slip_remote_address** parameter.
- The **slip_load_dump_host** parameter is used in place of the **pref_load_host** Annex parameter if the port is defined with the **load_dump_sequence** Annex parameter. In order to use a SLIP link for down loading software, you must specify the **slip_load_dump_host** parameter.
- Set the **load_dump_sequence** Annex parameter to define the port number(s) over which a boot or dump is to occur. The default is **net** for the local network, such as Ethernet. Enter a SLIP link as **sl***nn*, where *nn* is the port number. For example, if you want a boot request to try the SLIP link on port 2 first, then the SLIP link on port 6, and finally the local network, you would enter:

```
command: set annex load_dump_sequence sl2, sl6, net
```

- You can select up to four interfaces. The preference of the boot and/or dump request would be performed in the order you specified. Each interface must be separated by a comma.
- The preferred load host is defined by the network interface that is being used for the boot: the **pref_load_host** parameter applies when using the local network interface; the **slip_load_dump_host** parameter applies to individual SLIP links.
- The **slip_allow_dump** parameter disables using the SLIP link for dumps. If you include this port in the load-dump sequence list, the Annex automatically dumps to the host specified in the **slip_load_dump_host** parameter.

  On a line at 9600 baud, a dump of a 1 Mbyte Annex can take about ten minutes.

To download the operational code, but prevent dumps:

- Specify the port with the **load_dump_sequence** parameter.
- Specify the Internet address of the preferred load host with the **slip_load_dump_host** parameter.
- Set the **slip_allow_dump** parameter to **N**.

SLIP Configurations for Dial-up

Setting a port to provide dial-up SLIP requires that you configure the port for both an incoming modem and a SLIP link.

- Use the supplied defaults for the **data_bits** (8), **stop_bits** (1), and **parity** (none) parameters. Set the **speed** parameter to the rate required for the SLIP link. Set the remote end to match these values.
- Setting the **control_lines** parameter to **modem_control** causes the port to be reset when the modem hangs up. Otherwise, issuing an **na reset** command turns off the SLIP interface so that another call can be received.
- Set the **mode** parameter to **cli**.
- Set the **type** parameter to **dial_in**.

Note: The **cli_security** parameter enables CLI security for a dial-in SLIP line.

- The **slip_do_compression** parameter forces TCP header compression on the SLIP link. When enabled, the Annex always sends compressed packets and the other end of the SLIP link also must send compressed packets.
- The **slip_allow_compression** parameter allows the Annex compress TCP headers if the other end of the SLIP link also provides compression. When enabled, and the other end of the SLIP link sends compressed packets, the Annex sends compressed packets. If the other end of the SLIP link cannot compress packets, the Annex sends normal SLIP packets.
- The **slip_no_icmp** parameter discards any ICMP pakets that are destined for the SLIP link, thereby reducing unnecessary traffic and messages over the SLIP link.
- The **slip_tos** parameter causes the Annex to send interactive traffic (**telnet, rlogin,** and ftp control sessions) before any other traffic over the SLIP link; it provides a type-of-service queuing on the link.

Before issuing the **slip** command at the CLI connection, you must configure these SLIP parameters:

- The **slip_local_address** parameter defines the Internet address for this port. This address is the address for the Annex's side of the point-to-point SLIP link.
- The **slip_remote_address** parameter defines the Internet address for the remote end of this SLIP link. This is the address that must be used by the remote host/PC in order to function as a host on the LAN. The address must be unique for the network, but it can be assigned arbitrarily.

Dynamic Dial-up SLIP Address Assignment

Dynamically assigned SLIP remote endpoint addressing allows you to change the remote endpoint address for dial-up SLIP. The feature has certain requirements (CLI security enabled, no Annex password set) and involves some *C* programming in the ACP Annex security utility. Only an experienced *C* programmer should try implementing this feature.

To activate ACP dial-up:

In **src/erpcd/acp_policy.c**, add:

```
# define DIALUP_SLIP
# define USER_VALIDATION 1
```

Also in **src/erpcd/acp_policy.c**, write the **lookup_userport()** function. This function translates a given user Annex/port to an Internet address. Write this however you like, but adhere to the entry and exit conditions documented in the **lookup_userport()** module.

In **src/erpcd/makefile**, add:

```
../netadm/netadm.a
```

to both the **erpcd** dependency and build lines.

Using **na** or CLI **admin**, set the following parameters on the port of the Annex on which you want dial-up SLIP to run:

```
set annex enable_security y password ""
set port cli_security y
reset annex security
reset <port number>
```

Routing Across a SLIP Link

Although the Annex updates its routing table based on response messages broadcast by hosts' **routed** daemons and on ICMP redirects, it is a passive gateway and does not broadcast response messages itself. This means that the Annex with the SLIP interface routes packets addressed to the host at the remote end of the connection, but does not inform other hosts or Annexes that it has this capability. Other hosts and Annexes on the same network must be told about the route before they can use it.

Routing Between Two Networks

To make other hosts aware of a route over a SLIP link, use active routing in which a host running **routed** advertises a route for the Annex. Create an entry in a host's **/etc/gateways** file. Using the example in Figure B–4, *host03*, whose Internet address is 132.245.10.9, has the following **/etc/gateways** file entry:

```
host 132.245.5.9 gateway 132.245.10.7 metric 1 passive
```

This entry advertises a route for the host with the Internet address 132.245.5.9 through the Annex at 132.245.10.7. A host running **gated** can accomplish the same thing.

Having a host advertise a route results in an *extra-hop* situation. Hosts must direct their traffic destined for host 132.245.5.9 to host 132.245.10.9, which then routes the traffic to the Annex at 132.245.10.7. To avoid this extra hop, the host at 132.245.10.9 needs to send out an ICMP redirect message.

To make Annexes aware of a route using a SLIP link, create entries in the Annex **/usr/spool/erpcd/bfs/gateways** file. Using Figure B–3, the following would be the entries for Annexes on network 132.245.10:

```
annex 132.245.10.7

   net 132.245.5.0 gateway 132.245.99.2 metric 1 hardwired

else

   net 132.245.5.0 gateway 132.245.10.7 metric 2 hardwired
   host 132.245.99.2 gateway 132.245.10.7 metric 2 hardwired

end
```

These entries inform *annex01* that *host01* is a gateway to network 132.245.5 and inform other Annexes on network 132.245.10 that *annex01* is a gateway to either *host01* or network 132.245.5.

Extending a Single Host onto the Network

The Annex can use Proxy-ARP to extend a single host and remote Annexes onto the network transparently. A Proxy-ARP causes the Annex to answer ARP requests for the destination address of a SLIP link with its own hardware address. The following is an example of the type of ARP entry that would be used on the Annex for the SLIP interface in Figure B–5:

```
bunky (132.245.5.18) at 00-80-2d-00-26-cd permanent published
```

Typically, a Proxy-ARP is used when the Annex's SLIP link is to a single device, i.e., both the device and the Annex use the same Internet network address. No other routing information is required with this configuration (the **arp** command section provides for more information on manipulating the ARP cache.)

Chapter 7

Configuring Hosts and Servers

General

This chapter describes configuration requirements for hosts that provide the following services to the Annex:

- Accessing 4.2BSD hosts
- Setting up file servers
- Dump host services
- Host-based security services
- Setting up name servers
- Setting up a host for 4.3BSD syslogging

Accessing 4.2BSD Hosts

The 4.2BSD version of the **rlogin** protocol allows logins only from hosts whose names and Internet addresses are listed in the host's **/etc/hosts** file. The 4.3BSD version of the protocol does not impose this restriction.

Edit the **/etc/hosts** file on each 4.2BSD host that users need to access using the Annex **rlogin** command and add an entry for the Annex. Add the new entry near the beginning of the file because UNIX software searches this file sequentially.

Setting Up the File Server

Setting up a file server for the Annex involves installing the Annex source code onto a host. This is described in the *Annex Communications Server Installation Notes*. An installation script, called **install-annex**, is supplied with the software distribution. This script asks several questions about the host system, and then creates the files */annex_root*/**src/make.config** and */annex_root*/**src/Makefile**, where *annex_root* is the directory to which the Annex's source code is copied, to compile and install the support utilities in the directory you specify during the installation.

The following files are maintained on an Annex file server host in **/usr/spool/erpcd/bfs**: **gateways**, **rotaries**, **macros**, and message-of-the-day. Only the image file is required on the file server host. If the other files are not available on the host, the Annex broadcasts for them. You can disable broadcasting by setting the **load_broadcast** parameter to **N**.

Multiple Server Hosts

Using multiple file and security servers increases the Annex's reliability and availability. The Annex broadcasts requests for a download of operational code if the defined preferred load host does not respond to a boot request. If another host does not have the Annex's file server utilities, the Annex cannot boot. If an Annex cannot reach a security server, user requests that require authorization are rejected unless the local password protection is configured for back-up security.

To install file server software on multiple hosts, repeat the installation procedure on each host that will be a file and/or a security server.

If you are defining multiple security servers, the contents of the **acp_passwd**, **acp_keys**, and **acp_restrict** files must be identical on all security servers.

At the hosts on which you installed the Annex software, verify that the following line is in the **/etc/services** file (if not, add it):

```
erpc 121/udp            # Annex erpc
```

Examine the **eservices** file; make sure the right entry is not commented out (**bfs** or **acp**) for the type of server this host is providing.

Note: Ensuring that every host receives the latest updates to the Annex operational code prevents an Annex from loading an outdated version when booted.

Setting Up an Annex as a Boot Server

An Annex can be configured as a boot server and can supply a copy of its operational code and/or various files normally supplied during a boot for other Annexes. An Annex can provide operational code only for another Annex that shares the same operational code image.

When an Annex boots, it uses the image file to load the operational code, the **gateways** file to initialize the routing table, and the **rotaries** file to initialize the rotaries. The Annex does not store these files, because they use memory (the CLI **stats** command displays the amount of free memory). A file server host Annex uses approximately 56K bytes of memory for the operational code; for the **gateways, rotaries, macros** and message-of-the-day files, it uses the amount of host disk space from which these files are read.

To configure an Annex as a boot server, set the **server_capability** Annex parameter to one of these values (the default is **none**):

all	Copies of the operational code, and the **gateways, rotaries, macros,** and message-of-the-day files.
gateways	A copy of the **gateways** file.
image	A copy of its operational code.
macros	A copy of the **macros** file.
motd	A copy of the message-of-the-day file.
none	The Annex is not a file server.
rotaries	A copy of the **rotaries** file.

If you configure an Annex to supply only a copy of the operational code, the default is for the Annexes being booted to broadcast for the **gateways, rotaries,** message-of-the-day, and **macros** files. You can disable broadcasting for these files by setting the Annex **load_broadcast** parameter to **N**.

If you install new Annex software and/or if you change the rotaries or the gateways files, first reboot all Annexes that are boot servers before rebooting any other Annex on the network.

Setting Up the Message-of-the-Day File

The Annex can optionally display a message-of-the-day at the CLI. This message is displayed after the Annex has been rebooted or reset, after the port has been reset, or when a user accesses a virtual CLI.

To use this option, create an ASCII file on a file server host with the desired message. This file must be located in the **/usr/spool/erpcd/bfs** directory. The default file name is **motd**. If you elect to use another name for this file, you must specify this name with the **motd_file** Annex parameter.

The Annex reads the message-of-the-day file from the file server host each time it boots, and when the **na reset annex motd** command is issued. Initially, the Annex requests the file from the preferred load host. If that host is not defined or available, the Annex broadcasts a request for this file unless you disable broadcasting with the **load_broadcast** Annex parameters.

Installing a Time Server

The Annex host software includes a time server program because timer service is required for correct Annex operation. Time servers are available on many UNIX hosts through the **inetd** daemon. To determine if a specific system is running a time server, use the UNIX **netstat** command with either the **-a** or the **-a** and **-n** options. A time server is displayed as listening on UDP port **time** (or **37**).

If a host time server is not present, use the supplied **timserver** program. Perform the following steps to start the program:

1. Add the following line to **/etc/services** if not already included:

   ```
   time 37/udp timeserver
   ```

2. Start the server, e.g.:

   ```
   # /etc/timeserver
   ```

3. Edit the appropriate **rc** file so that the time server starts automatically when the system is booted.

By default, the Annex does not broadcast for the time. You can enable broadcasting for a time server using the **time_broadcast** Annex parameter.

Routing Services and the gateways File

The Annex manages its network routing tables with a routing daemon similar to 4.3BSD **routed**. These routing tables maintain information on gateways that provide access to hosts on different networks. The Annex builds this table dynamically by monitoring ICMP redirects and Routing Information Protocol (RIP) broadcast messages. The Annex updates the table based on the information received from the redirects and broadcast messages (the CLI **netstat -r** command displays the routing table). The Annex provides two options for maintaining the routing table:

- Create a **gateways** file for the Annex in which you can define fixed routes to a destination.
- Disable the RIP-listener on the Annex causing it to rely only on the **gateways** file.

Annex gateways File

The Annex's **gateways** file resides on the file server host. Using this file, you can define a route to a destination (either hosts or other networks) as active, passive, or hardwired (fixed).

An active gateway can generate RIP update messages. If this gateway does not transmit these broadcasts on a regular basis, it eventually is deleted from the routing table. If another route to the destination associated with the gateway is received, the new route replaces the entry for the active gateway.

A passive gateway cannot generate RIP messages, and therefore, is never removed from the routing table. If an alternate route is received for the destination, the entry in the routing table is updated.

A hardwired gateway has a fixed route to the destination. This route cannot be changed or deleted, even if a routing update is received.

Creating the gateways File

The Annex **gateways** file is an ASCII file that can be created using any text editor. The file's full pathname is **/usr/spool/erpcd/bfs/gateways**. After an Annex boots, it downloads the file from the preferred load host. If the Annex does not receive a **gateways** file from the preferred load host, it broadcasts for its **gateways** file unless you disable broadcasting using the Annex **load_broadcast** parameter. If the Annex does not locate a **gateways** file, it assumes there is none on the network.

The **gateways** file also is used to configure routes for SLIP links (see *Chapter 6: Serial Line Internet Protocol (SLIP)* in this book) and enable the Simple Network Management Protocol (SNMP) agent (see *Book C: Network Management, Chapter 2: Simple Network Management Protocol (SNMP)*). The definitions in the **gateways** file must conform to the following conventions:

- A pound sign (#) in the first column starts a comment. The comment is terminated by the end-of-line.
- Non-delimited white space (i.e., spaces, tabs, etc.) are treated as a single space.
- All keywords and port information are case-dependent.

The format for an entry in **/usr/spool/erpcd/bfs/gateways** is:

{net|host} *addr1* **gateway** *addr2* **metric** *hops* **{active|passive|hardwired}**

The fields are:

net|host	Indicates whether destination specified with *addr1* is either a network or a host.
addr1	The Internet address of the destination. Use either 0 or default to add a default route.
addr2	The Internet address of the gateway that reaches the destination address.
hops	Indicates the number of hops needed to reach the destination.
active	Indicates that the gateway is an active gateway capable of generating routing updates.
passive	Indicates that the gateway does not generate its own routing information.
hardwired	Indicates that the route through this gateway is fixed.

The **gateways** file allows you to define lines in the file that refer only to an Annex. The syntax for the extension is:

annex *ipaddr*
...
end

The lines enclosed by the **annex...end** pair are to be used only by the Annex with the Internet address *ipaddr*. An **else** keyword also can be used (alone on a line) to list configuration information for all Annexes, except the one identified on the **annex** line. You cannot nest **annex...end** entries. Following is an example of this extension to the **gateways** file. This example shows how to configure a SLIP interface (see Figure B–4 in *Chapter 6: Serial Line Internet Protocol (SLIP)*).

```
# SLIP link to the 132.245.5 net
annex 132.245.10.7

   # 132.245.10.7 is a gateway to the entire 132.245.5 net
   net 132.245.5.0 gateway 132.245.10.7 metric 1 hardwired

else

   # other Annexes will route to 132.245.5 via 132.245.10.7
   net 132.245.5.0 gateway 132.245.10.7 metric 2 hardwired

end
```

The Annex logs *critical* level event messages for format errors or missing information in the **gateways** file if **syslog** is enabled.

Host Table Initial Loading in the gateways File

When the Annex boots, it adds host name entries to the host table. These host table entries will *live* in the host table until a nameserver overrides the entry's information or until the administrator resets the Annex nameserver via the **na** or CLI **admin** commands. These entries are similar to the **/etc/hosts** file entries, except aliasing is not supported.

A host name entry is an entry with an Internet Address (in decimal dot notation) followed by white space (blanks and/or tabs) followed by a host name (the host name may not contain blanks, tabs, or newlines). Some examples of host name entries are:

192.9.200.1 cbrown
192.9.200.2 snoopy
192.9.200.3 linus
192.9.200.4 lucy
192.9.200.5 sally
192.9.200.6 schroeder

Host name entries may be conditional with the use of the *annex* and *end* statements. The address on the *annex* statement ***must*** be an Internet Address. (For more information on the host table, see *Book C: Network Management, Chapter 1: Network Administration, Managing the Host Table.*)

Disabling the RIP-Listener

Another option for maintaining routing tables is to disable the RIP-listener at the Annex by setting the **routed** Annex parameter to **N**. This prevents the Annex from reacting to RIP broadcasts and using alternate routes. The Annex will construct its routing table solely from the information contained in the **gateways** file.

If you disable the RIP-listener, define a default route to a smart gateway in the Annex's **gateways** file. Specify the destination Internet address for the default route using either a **0** or the word **default**. This gateway sends ICMP redirects to the Annex, allowing it to learn only the required routes.

Customizing the User Interface

The Annex provides a macros feature for customizing the CLI user interface. This feature includes the ability to create aliases and menus that can be executed at the CLI or when accessing a port through the port server. The macros feature is contained in an ASCII file that is downloaded to the Annex. Both the CLI and the **na** program provide commands that assist you in managing these features.

Using the macros feature, you can set up site-specific prompts, commands, and menus. Aliases can make the Annex CLI invisible to the user. Menus hide the Annex's command interface and at the same time provide user options. Aliases and menus can be bound to specific ports on an Annex. Also, aliases can be created for slave ports that are accessed through the port server. Individual aliases and menus can be configured to display each time a port is accessed.

Creating the macros File

The file that contains the macros is **/usr/spool/erpcd/bfs/macros**. The Annex loads the **macros** file when it boots or when the **na** command **reset annex macros** is issued. If the preferred load host is not available, the Annex broadcasts for this file, if the Annex **load_broadcast** parameter is set to **Y**. The definitions in the file must conform to the following conventions:

- A pound sign (**#**) in the first column starts a comment. The comment is terminated by the end-of-line.
- Non-delimited white space (i.e., spaces, tabs, etc) are treated as a single space.
- All keywords and port information are case dependent.
- Many strings, including description strings, must be enclosed in delimiters. The delimiters may be any printable character not contained within the string.

The basic entries in the **macros** file are of one of the following forms:

alias |*description*|
keyword arguments
{
alias expansion
}

menu |*description*|
keyword arguments
{
menu expansion
}

The **alias** begins an alias definition; the **menu** begins a menu definition.

The *description* is a string, which may contain spaces. The string is what is displayed by the CLI **help** command for the alias or menu.

The bar character (|) around the *description* is the string delimiter; the delimiter can be any character.

The argument *port_set* is a list of ports that applies to the entry. A *port_set* can be specified using one of the following formats:

- A serial port number
- A range of serial port numbers separated by a dash
- A **v** designating virtual CLI connections
- A list of any of the above separated by commas

The whole *port_set* can be followed by an @ and the name or Internet address of an Annex. If you do not specify an Annex for a *port_set*, the macro applies to the specified ports for any Annex that reads the **macros** file. The *port_set* cannot contain any spaces. If a *port_set* is not defined for a menu or alias, the entry applies to all ports on the Annex reading the file and all virtual CLI connections. Examples of *port_set* entries are:

```
5@132.245.6.42
1,3,5,v@132.245.6.78
1-13@thirdfloor
1,3,5,v
1-13
```

The supported keywords are:

keyin |*name*| *port_set*	Binds *name* to current macro on the ports defined in *port_set*. The *name* is used to execute the **menu** or **alias**. The *name* must be delimited.
init_cli *port_set*	Binds the macro to be executed on the initialization of a CLI at any port in the *port_set*. Applicable only with CLI or virtual CLI ports.
init_psrv *port_set*	Binds the macro to be executed when the port server connects to any port in the *port_set*. Applicable only with ports whose **mode** parameter is set to **slave** or **adaptive**.
cmd_list *cmd1,cmd2,...*	Specifies the commands that are available with the menu being defined. Each command is separated by a comma. A command can be a macro (**menu** or **alias**) or a CLI command. Spaces are not permitted in the command list. Applicable only with menus.

Note: Aliases listed in a **cmd_list** must be valid for the same ports as defined with the **keyin** keyword that defined the alias.

Following the keyword is the expansion text of the macro. This text consists of a series of lines with one entry per line. The expansion text begins after a line with a single open brace ({) and ends before a line with a single close brace (}). If the expansion text is for an alias, it contains an *alias expansion*; for a menu, the expansion text contains a *menu expansion.* An *alias expansion* permits the following statements:

> *string*	Indicates that the *string* should be transmitted to the serial port; for a virtual CLI, it directs the *string* to the connected device. This can be used to display a message to the CLI use. The *string* is not delimited.
< *string*	Simulates the *string* being input from the serial port; for a virtual CLI, it simulates input to the CLI.
< pause	Indicates a pseudo-CLI command that causes the macro to display *Hit any key to continue* on the serial port and to wait for a key stroke before continuing. This is applicable only with alias definitions used on a CLI.

The *menu expansion* defines the menu that is displayed. Each line of the *menu expansion* is a separate line of the menu. If no lines are defined between the open and close braces, the Annex creates a generic menu containing a list of the commands that were defined with the **cmd_list** keyword and their descriptions. If you include any superuser CLI commands in a menu, they display with the menu, but users can access the commands only through the CLI **su** command.

Examples of Aliases and Menus

The following sample **macros** file defines a series of aliases combined into a menu. The menu is accessible from ports 1 through 6, ports 10 through 16, and all virtual CLI connections for an Annex at a given address.

```
########################################################################
#
#
# Set up a macro for annex at <annex-address>
#
# This Macro sets up a menu for port 3 and all virtual ports
# on the specified annex.
#
# Other examples of setups are:
#
#  keyin "#" 1-64@<annex-address>
#  ( 1-64@address = ports 1 through 64 for annex @ address)
#
#  keyin "#" v@<annex-address>
#  ( v@address    = all virtual ports for annex @ address)
#
#  keyin "#" v
#  ( v            = all virtual ports for any annex that
#                   boots from this host)
#
#  keyin "#" 1-64
#  ( 1-64         = ports 1 through 64 for any annex that
#                   boots from this host )
#
#
# The menu displays after clearing the screen, five lines down from the
# top of the screen and 27 spaces over.
#
# This macro limits the possible commands at the Annex prompt to only
# those listed on the menu.
#
# This macro file does not affect any other Annex that boots from this
# Unix host.
#
# All other ports on the Annex have full access to cli commands.
#
# Note:  Replace "<annex-address>" with your Annex's internet address
#        (e.g. 192.9.200.1).
#
#        Replace "<system-address>" with your system's name or internet
#        address (e.g. fred or 192.9.200.2).
########################################################################
#
# Rlogin to system1
#
alias "Connect to System1"
    keyin "1" 3,v@<annex-address>
{
<rlogin <system1-address>
}
```

```
#
# Rlogin to system2
#
alias "Connect to System2"
    keyin "2" 3,v@<annex-address>
{
<rlogin <system2-address>
}

#
# Issue a "who" command to determine who is running on the Annex.
#
alias "Who?"
    keyin "3" 3,v@<annex-address>
{
<who
>
<pause
}

#
# Do hangup from Annex port. This disconnects the Annex port.
#
alias "Exit"
    keyin "4" 3,v@<annex-address>
{
<hangup
}

#
# This section defines the actual menu.
#
menu |Generic Menu Header|
    init_cli 3,v@<annex-address>
    keyin "menu" 3,v@<annex-address>
    cmd_list 1,2,3,4
{
#
# The "^[[2J" is a control sequence of
# CTRL-[ followed by "[2J"    (clear the terminal screen)
# CTRL-[ followed by "[5;27H" (go to line 5, space over 27 spaces)
# CTRL-[ followed by "[9;15H" (go to line 9, space over 15 spaces)
#
^[[2J
^[[5;27HGeneric Menu Header
^[[9;15H1)    Connect to System1
^[[11;15H2)   Connect to System2
^[[13;15H3)   Who?
^[[15;15H4)   Exit
^[[17;15H^[[16pEnter Number: ^[[Op
}
```

For the aliases in the previous example to work, CLI security must be enabled for the ports in the *port_set*, and the destination host must trust the Annex by its inclusion in the host's **/etc/hosts.equiv** file. Since the user was authenticated using Annex security, **rlogin** passed the user name and the password was not required.

The last entry creates the display for the menu and includes the above aliases in the menu's **cmd_list** as well as the CLI **help** command. This entry also includes an **init_cli** keyword, which causes the menu to be initialized each time the port or the virtual CLI is reset. This places the user into the menu without communicating with the CLI. The menu includes ANSI terminal control codes to set up the menu display. For example:

```
          Generic Menu Header

    1)    Connect to System1
    2)    Connect to System2
    3)    Who?
    4)    Exit

    Enter Number:
```

The following example uses the same aliases as in the previous example, but does not provide a *menu expansion* to define the menu display:

```
menu |Annex menu|
    init_cli 3,v@<annex-address>
    keyin "menu" 3,v@<annex-address>
    cmd_list 1,2,3,4
{
}
```

This entry creates the following menu:

```
          Annex Menu

1     : Connect to System1
2     : Connect to System2
3     : Who?
4     : Exit
```

The following sample macro file automatically connects any user logging in on ports 1–32 of the defined Annex to the given *<system-address>*, without requiring a keystroke; the virtual ports have normal connection options. This macro is both Annex- and host-specific.

Note: Set the **cli_inactivity** parameter to **immediate**: when a user logs off the last job, the macro is re-initiated; otherwise, the CLI prompt returns at logout.

```
alias "Connecting to host"
    init_cli 1-32@<annex-address>
{
>
> Please wait while you are connected......
>
<rlogin <system-address>
}
```

Managing the Macro Feature

The CLI **help** command displays an alias name and description with other valid CLI commands. When in superuser mode, the CLI **help** command supports the **-m** argument, which displays a list of all macros and their assigned *port_set* defined for that Annex. For example:

```
annex01# help -m

Name    Assigned Ports              Description
=============================================================
1       1-6,10-16,v                :Connect to Accounting
2       1-6,10-16,v                :Read EMAIL
3       1-6,10-16,v                :Show users on the network
4       1-6,10-16,v                :Exit system
annex01#
```

If you include the name defined with the **keyin** keyword after the **-m** argument, the **help** command displays the definition defined with that entry. For example:

```
annex01# help -m 2

Macro Name: 2       Description: Read EMAIL
Assigned Ports: 1-6,10-16,v
Functional Text:

<rlogin maildrop
<mail

<<
annex01#
```

The **na** command **reset annex macros** causes the Annex to reload the **macros** file. This provides you with the option of modifying the **macros** file and loading it onto an Annex without having to reboot.

Dump Files

The Annex can dump its memory image to a dump host on demand through either the superuser CLI **boot -d** command or the **na dumpboot** command or on certain software or hardware events. A non-recoverable hardware or software error triggers Annex dumps. Dump files are intended for use by technical support personnel only.

The host to which an Annex sends a dump must be running Annex file server software. For the Annex, you can define a preferred dump host, which the Annex first tries to up load a dump file. If this address is not specified, the Annex broadcasts a request and dumps to the first host that responds.

At the dump host, the dump creates a file (between one and four Mbytes in size) in the directory **/usr/spool/erpcd/bfs** and assigns a unique dump file name to each Annex. The assigned name depends on whether the dump host can support file names longer than 14 characters.

On hosts that support file names longer than 14 characters (for example, BSD UNIX hosts), dump files are named **dump.***xxx.xxx.xxx.xxx*. The file extensions *xxx.xxx.xxx.xxx* are the Annex's Internet address.

On hosts that limit file names to 14 characters (for example, System V hosts), the dump creates two additional directories under **/usr/spool/erpcd/bfs**. The name of the first directory is **dump**; the second is the Annex's Internet network address. (Note, subnet addresses are not used in naming the dump file.) The name of the dump file is the Annex's Internet host address. Table B–2 gives examples of dump file names (all pathnames are relative to **/usr/spool/erpcd/bfs**).

Table B–2. Dump File Naming Conventions

Annex Address	Network Address	BSD Filename	System V Pathname
63.0.0.75	63	dump.63.0.0.75	dump/63/0.0.75
131.14.23.1	131.14	dump.131.14.23.1	dump/131.14/23.1
195.46.2.15	195.46.2	dump.195.46.2.15	dump/195.46.2/15

Host-based Security

The Access Control Protocol (ACP) provides the following host-based security options:

- User validation for connecting to a CLI.
- User validation for access to the port server.
- Validation for host connections.
- Logging security events.
- Encryption key protection for Annex access to the security server, and optionally, for access to an Annex through **na**.

ACP requires that at least one host is a security server and that security is enabled on the Annex. All files that maintain access validation reside on the security server.

The Annex software supplies a policy application that provides a model on how host-based security functions. With host-based security, the Annex logs messages to the security server when security events occur and when the Annex is booted. You can modify this supplied policy to:

- Disable user name and password validation.
- Change the name of the password file.
- Disable CLI commands.

Using the Annex distribution's library calls, and any library calls available on your system, you can modify the code to provide your own prompts and encoded validation.

Defining a Security Server

The ACP security server software is provided as part of the expedited remote procedure call daemon (**erpcd**) supplied with the file server software. Included with the software is the **eservices** file that has two entries: one for the block file server (BFS) and one for Access Control Protocol (ACP).

The file that is supplied with the Annex software distribution has the ACP service commented out, as follows:

```
# erpc remote programs
#
# prog no.   verlo     verhi      name
#
1            0         99         bfs
# 3          0         99         acp
```

To define a security server, you must install the file server software on a host and delete the # symbol in front of the ACP entry. For example:

```
# erpc remote programs
#
# prog no.  verlo      verhi      name
#
1           0          99         bfs
3           0          99         acp
```

You can use the same host to be both file server and security server, or move the servers to two separate hosts. You can put the security server on more than one host. However, you must define one host as a first preferred security server. You can optionally define a second host as a second preferred security server. Additional security servers would respond to broadcast requests if the preferred servers are unavailable.

Note: The contents of the files **acp_passwd**, **acp_restrict**, and **acp_keys** should be identical on all servers.

Creating User Password Files

CLI ports are configured for user validation with the **cli_security** port parameter. Virtual CLI connection user validation is enabled with the **vcli_security** Annex parameter. User validation on accessing ports through the port server is provided with the **port_server_security** port parameter.

Note: User validation is available only if host-based security is enabled at the Annex using the **enable_security** Annex parameter.

For user validation with the supplied security policy, the Annex sends a request to the security server to compare a user name and password against entries in a password file. If a match is found, the user is granted access to the CLI; otherwise, the user is denied access to the CLI.

The supplied policy expects the password file to be the **acp_passwd** file. This file uses the same format as the **/etc/passwd** file. The easiest way to create this file is to copy the **/etc/passwd** file to **acp_passwd**. One advantage for creating the **acp_passwd** file this way is you can merge **/etc/passwd** files from different hosts into one file on the security server. This allows you to create a network-wide password file. The following is an example of the supplied user validation with CLI security:

```
Annex Command Line Interpreter * Copyright 1990 Xylogics, Inc.

Checking authorization, Please wait...
Annex username: kathryn
Annex password:

Permission granted
annex:
```

Setting Up Connection Security

ACP provides connection security, which authorizes connection requests to hosts using the CLI connection commands **telnet** and **rlogin**. The connection security mechanism uses a host-resident file that lists the hosts to which connections are restricted.

Connection security is enabled for individual ports using the **connect_security** port parameter. Connection security requires that security be enabled for the Annex using the **enable_security** Annex parameter, which automatically enables connection security for all virtual CLI connections. When a user issues a connection command, the Annex checks a restrict file for permission to connect to that host. The supplied policy expects the restrict file to be **acp_restrict**, which is an ASCII file that you create with any text editor. The entry format is:

annex: restricted host, restricted host, ...
annex~ unrestricted host, unrestricted host, ...

Entries in this file include:

annex	The name or Internet address of the Annex for which connect security is enabled.

: (colon)	Indicates the hosts listed in the same entry are restricted.
~ (tilde)	Indicates the hosts are unrestricted.
restricted host	The name or Internet address of a restricted host (including Annexes). The list of restricted hosts is separated by commas.
unrestricted host	The name or Internet address of an unrestricted host (including Annexes). The list of unrestricted hosts is separated by commas.

An asterisk (*) can be used as a wild card in place of a host name or the host part of an Internet address. Following is an example of two restricted-host entries:

```
annex01: hosta, hostb, hostf, 132.245.6.23
annex02: hostc, hoste, 132.245.6.15, hostf, 132.245.6.23, hosth, annex01
```

The first entry prevents users in which connection security is enforced on *annex01* from accessing *hosta*, *hostb*, *hostf*, or the Internet address 132.245.6.23. The second entry prevents similar users on *annex02* from accessing six hosts and one Annex. Connection security is enforced on all virtual CLI connections at *annex01* and *annex02*.

Hosts that are not listed in the file are considered unrestricted. Since ACP searches the **acp_restrict** file sequentially, the order of placement in the file is important. The search stops when it finds a host that matches. You can use unrestricted host entries to prevent users on one network from accessing hosts on any other network. In the following example, the policy finds the unrestricted definition for Annexes and hosts on network 192.17.5 and grants access. It finds the restricted definition for hosts on any other network and does not grant access.

```
192.17.5.*~ 192.17.5.*
192.17.5.*: *
```

The next example illustrates wild cards. Here, a *publichost* is defined as accessible from all Annexes and a *securehost* is inaccessible from any Annex.

```
*~ publichost
*: securehost
```

If permission is granted to a connection security request, the user follows the normal login procedure. If the request is denied, the message *Permission denied* is displayed and the session (job) is aborted.

Creating the ACP Encryption Keys File

The host-based security can encrypt messages between an Annex and the security server. Encryption of messages is based on the ACP key, which is set with the **acp_keys** Annex parameter. This option is available when the host-based security is enabled for the Annex with the **enable_security** Annex parameter.

When ACP encryption is enabled, the supplied policy expects a file called **acp_keys**, which maintains a list of Annexes and their respective encryption key. When the security server receives an encrypted message from an Annex, the server tries to match that key against the key assigned to the Annex in the file. If no match exist, the Annex and the server cannot communicate.

Each entry in the encryption keys file contains a list of Annex names or Internet addresses separated by commas and an encryption key for those Annexes. The Annex or the list of Annexes and the key are separated by a colon. The order of placement in the file is important, as the file is sequentially read. Syntax rules for the **acp_keys** file are:

- Any part of an Internet address in the list can be specified with an asterisk (*).
- A backslash (\) is used to escape a new line.
- Any ASCII character except spaces and tabs are valid encryption keys.
- Each key can contain a maximum of fifteen characters.

Typically, Annexes that do not expect encryption do not need to be included in the file. However, if an Annex falls into a group of multiple Annexes that were specified with an * as part of the Internet address and it is not expecting encryption, that Annex must be included in the file with the key left blank.

In the following example, the first three entries specify *insomniac-1* as the key for the Annex whose Internet address is 132.245.6.15, no encryption for the Annex whose Internet address in 132.245.6.75, and *Piano* as the key for all other Annex on the same network. The last entry specifies *gl12ch* as the key for *annex01*, *annex02*, and *annex03*. Each **acp_key** parameter for the Annexes listed in the example must be identical to the key included in the **acp_keys** file.

```
132.245.6.15: insomniac-1
132.245.6.75:
132.245.6.*: Piano
annex01, annenx02, annex03: gl12ch
```

Changing the value of the **acp_key** parameter on any Annex requires the same change to the **acp_keys** file on the security server. The recommended order for changing the ACP encryption key on an Annex is:

1. Edit the **acp_keys** file on all security server hosts.
2. Using **na**, change the value of the **acp_key** parameter for all affected Annexes.
3. Update the cache by sending the **erpcd** on all security server hosts a HUP signal with **kill**.
4. Reset the security subsystem for all affected Annexes using the **na** command **reset annex security**.

Logging Security Events

Host-based security provides the ability to generate audit trails of user activity. Each time the security server grants or denies a request for user access, the security server logs it. Each event is logged as a message to a file defined as **acp_logfile** (see *Book C: Network Management, Chapter 1: Network Administration*). You can change **acp_logfile** in the file */annex_root*/**src/erpcd/acp_policy.h** to any suitable pathname including **/dev/null**, **/etc/console**, etc.

Each logged message contains the Internet address of the Annex, a sequence number, the port number, the date and time, the event, and other information that is protocol dependent. The fields in the file are separated by colons and can be used by UNIX utilities that sort, merge, select, or filter streams.

Modifying the Supplied Security Application

You can modify the supplied security policy to create a security scheme that meets the needs of your network. Some simple modifications involve changing system definitions in the file */annex_root*/**src/erpcd/acp_policy.h**. More elaborate security policies may require modifying or replacing functions in the file */annex_root*/**src/erpcd/acp_policy.c**.

Note: Do **not** change the function declarations or the description of the interface; these are fixed by the calls made into this library. Before making even the smallest change, save the base version of the file requiring modification.

If you modify the default policy, you must recompile **erpcd**, kill the current version, and start the new version. Instructions for doing this are provided later in this section.

Disabling User Name and Password Validation

When security is enabled, users are requested to provide a user name and password. You can disable this policy by modifying the */annex_root*/**src/erpcd/acp_policy.h** file to change the line that defines user validation from:

```
#define USER_VALIDATION 1
```

to:

```
#define USER_VALIDATION 0
```

Messages are logged to the security server host when users access the CLI, but no user name is included in the message.

Changing the Expected Names of Files Used by ACP

The supplied policy uses supplied names for various files. For example: **acp_passwd**, **acp_keys**, **acp_restrict**, and **acp_logfile**. You can change the names of any of these files in the */annex_root*/**src/erpcd/acp_policy.h** file.

If you decide to use either an existing system or a network-wide password file instead of **acp_passwd** file, change the following line in the */annex_root*/**src/erpcd/acp_policy.h** file from:

```
#define ACP_PASSWD INSTALL_DIR/acp_passwd"
#define ACP_PTMP INSTALL_DIR/acp_ptmp"
```

to:

```
#define ACP_PASSWD "new_filename"
#define ACP PTMP "new_tempfile"
```

The *new_filename* is the name of the new password file, and the *new_tempfile* is a temporary file used by the **ch_passwd** command. Since you do not need the temporary file if you are using an existing system file, comment out the line for the temporary file.

Note: INSTALL_DIR is defined in the file */annex_root*/**src/make.config** with the leading quote supplied by the make file. Since the trailing quote is required by the two strings, double quote the names for the new password and temporary files.

Disabling CLI Commands

When the security subsystem is enabled, you can mask user access to specific CLI commands by changing the following line, which defines user commands as not masked, in the */annex_root*/**src/erpcd/acp_policy.h** file:

```
#define CLI_MASK (unsigned long 0)
```

To mask the **hosts, netstat,** and **stats** commands, modify that line to read:

```
#define CLI_MASK (MASK_HOSTS | MASK_NETSTAT | MASK_STATS)
```

If the user enters the masked command, CLI displays an error message. In addition, the masked command is not available with the CLI **help** command. The superuser CLI commands cannot be masked individually. They can all be disabled by masking the **su** command.

Modifying the Code

You can create a more elaborate security policy application by modifying the code in the files */annex_root*/**src/erpcd/acp_policy.c** and */annex_root*/**src/erpcd/acp_policy.h**. The program that executes ACP *forks* itself each time a security request is received from an Annex. A call is made to an ACP remote procedure, which makes calls to functions in the ACP library to prompt for user names, passwords, etc. When ACP gathers the information required to perform the authorization algorithm, it again calls functions in the library to grant or deny the request. The program then exits.

The distribution policy file */annex_root*/**src/erpcd/acp_policy.c** is documented in the form of *C* programming language comments. The file */annex_root*/**src/erpcd/policy.doc** provides a complete description of the available library functions.

Recompiling erpcd

You must recompile **erpcd** if you modify the supplied policy and the **ch_passwd** utility if you changed the name of the ACP password file from **acp_passwd**. The source files are in */annex_root*/**src/erpcd**, where *annex_root* is the directory to which the Annex's source code was copied.

To recompile:

1. **cd** to */annex_root*/**src/erpcd**.
2. To recompile only **erpcd**, enter the command:

```
# make -f ../make.config -f Makefile erpcd
```

3. To recompile both **erpcd** and **ch_passwd**, enter the command:

```
# make -f ../make.config -f Makefile all
```

4. To install, enter the command:

```
# make -f ../make.config -f Makefile install
```

   This saves the old version of **erpcd** as **OLDerpcd** in the installation directory.

5. Kill the current **erpcd** and start the new one.

Setting Up Name Servers

The Annex uses various means of creating and maintaining the host table. This table includes the host names and the corresponding Internet addresses of hosts known to the Annex. The host table is generated by querying a name server and/or listening for broadcasts from RWHO daemons running on other hosts. The Annex adds entries to the host table when it receives:

- RWHO broadcast messages from other hosts.
- Responses from the Domain Name System (DNS) server and/or the IEN-116 name server to a query for an Internet address.

Annex parameters allow you to configure an Annex to listen only for RWHO broadcasts or to query one or both types of name servers or use both means of building a host table (see *Using Name Servers* in *Chapter 1: Configuring the Annex* in this book).

By default, the Annex builds the host table exclusively from RWHO broadcasts. Depending on what is available for your network, you may elect to use a name server in conjunction with RWHO broadcasts or disable the use of RWHO in the Annex.

If the network is using a domain name server, you must have a resource record for each Annex to the domain server. If the network does not have any name servers, the Annex distribution provides source for an IEN-116 server that you can install.

To determine if a specific system is running a name server, use the UNIX **netstat** command with the **-a** option or the **-a** and **-n** options. An IEN-116 name server is displayed as listening on UDP port **name** (or **42**). A BIND server is displayed as listening on UDP or TCP port **domain** (or **53**).

Adding the Annex to a Domain Name Server

The following discussion on adding a resource record for an Annex on a domain name server is specific to the Berkeley Internet Name Domain (BIND) server. If your network uses an alternate domain name server, see the documentation for that server.

For an Annex to obtain its name, you must include a PTR resource record for the Annex in the server's domain data files, specifically the IN-ADDR.ARPA domain. This record must contain the Annex's fully qualified domain name (FQDN), so the Annex can use part of the full domain name to expand host names to full domain names. The FQDN must always be supplied with a query to a DNS server; otherwise, the Annex adds one.

The following example shows a PTR resource record in a BIND name server for the Annex *annex01* with an Internet address of 132.245.6.34 and a full domain name of *annex01.eng.Widget.COM*:

```
34.6.245.132.IN-ADDR.ARPA. IN  PTR annex01.eng.Widget.COM.
```

After the Annex boots, it queries the name server for 34.6.245.132.IN-ADDR.ARPA. The name server returns the Annex's full domain name. The Annex uses the part of the name up to the first period as its name, and stores the rest as the default domain name. The Annex uses the default domain name to expand other host names when it queries the server for their Internet addresses. For example, to obtain an Internet address for the name *wayland*, the Annex sends a query to the name server for *wayland.eng.Widget.COM*. If the name server does not have this name, the Annex then queries for *wayland.Widget.COM*. If this query also fails, the Annex queries for *wayland*.

If you do not want the host name to be expanded with the default domain name, append a period to the host name. For example:

```
annex: rlogin wayland.
```

Installing the IEN-116 Name Server

Some UNIX-based systems include an IEN-116 name server, which can probably be used by the Annex. However, if an IEN-116 name server is not available on the network, the source code for a server is provided in the file */annex_root*/**src/ien-116**. To install the server:

1. Compile the source if necessary
2. Examine **/etc/services** and add the following line if necessary:

   ```
   name 42/udp nameserver          #IEN-116
   ```

3. Start the name server by entering:

   ```
   # /etc/ien116d
   ```

4. Edit the appropriate **rc** file so that the name server starts automatically when the system is booted.

To verify that the server responds to queries from the Annex, configure the Annex to use an IEN-116 server (see *Using Name Servers* section in *Chapter 1: Configuring the Annex* in this book). Look at the Annex host table. Then look at the name server host's **/etc/hosts** file and select a host that does not appear in the Annex host table. Using the CLI hosts command, force the Annex to query the name server.

Setting Up Hosts for Syslogging

The Annex provides a 4.3BSD system daemon for logging events to a host. If you log Annex events, set up the logging capabilities on the host. This host should be the one you specified using the Annex **syslog_host** parameter.

If the syslog host is a 4.3BSD system and you are using **syslogd**, you must define Annex logging by adding at least one line to the logging file **/etc/syslog.conf**. This line includes the name of the syslogging facility you specified using the Annex **syslog_facility** parameter, the priority level to which the host's syslogging daemon (**syslogd**) logs events, and the file to which Annex events are logged.

The priority level for this entry logs all events at that level and higher. Configure the Annex's priority levels using the Annex **syslog_mask** parameter. Although you can configure multiple priorities, you must select a level for this entry that logs all Annex events that you want to capture in the file. Regardless of the priority level you define, the Annex sends events to the host according to the priority defined in **syslog_mask**.

Following is an example of an entry for logging Annex events:

```
# Annex logging
local7.debug            /var/spool/log/annex
```

After creating an entry for Annex event logging, create the log file. Next, re-initialize the syslog daemon.

Configuring LAT Services

The Annex can display, and connect to, currently available LAT services. Initially, all LAT functions in the Annex are disabled: the user does not have access to the **connect** and **services** CLI commands, and the administrator does not have access to the LAT **na** parameters. To enable the LAT functions, the administrator must enter the correct value for the **na** parameter **lat_key** and reboot the Annex.

Note: When the administrator selects an Annex in **na** that has LAT enabled, **na** returns the normal configuration line with the string *W/LAT* included:

```
command: annex zippy
zippy: Annex-3-UX R6.0 W/LAT, 32 ports
```

The **lat_key** value is unique for each Annex. Since this value varies by port count, if the administrator changes the number of ports on the Annex, the **lat_key** must change and the Annex must be rebooted.

Note: Contact Xylogics to obtain your **lat_key** value.

The LAT-specific **na** parameters are: **server_name**, **facility_num**, **service_limit**, **keep_alive_timer**, **circuit_timer**, **retrans_limit**, and **group_value**.

Accessing LAT Services

The administrator must enable a set of the Annex's group codes that correspond to the site requirements. The Annex's group code is a security access mechanism designed to allow selective restriction of LAT services on the network. There are 256 group codes (0–255). Each group code is either enabled or disabled.

Each LAT service has an associated set of group codes. The users on an Annex will have access to a LAT service only if the service and the Annex have at least one enabled group code in common. For example, if the desired LAT service has group codes 1 and 3 enabled, the Annex must have either group code 1 or group code 3 enabled to access the service. If the Annex has only group code 0 enabled, the Annex users will not have access to the service.

The Annex maintains information only for the services to which its users have access; the **services** CLI command displays only the services to which Annex users have access. To change the status of the Annex's group codes, the administrator must change the **na** parameter **group_value**. The **group_value** field displays the group codes that are enabled; initially, the value is *none*. The administrator must determine which LAT services to enable, and set the group codes accordingly:

set annex group_value group_list *enable* | *disable*.

The **group_list** must be integers (0–255) separated by either a comma (,), to indicate multiple group codes, or a hyphen (-), to indicate a range of group codes; blanks are not permitted as delimiters. For example, **set annex group_value** *0–10, 15, 20, 240 enable* enables group codes 0 through 10, and group codes 15, 20, and 240.

For convenience, specifying *all* indicates all group codes. Thus, **set annex group_value** *all enable* enables group codes 0 through 255. The administrator can disable group codes in the same way using the keyword *disable* instead of *enable*.

After LAT is enabled, the appropriate group codes are enabled, and the LAT parameters have been reset (using the **reset annex lat** command), the **services** CLI command may show no services available for approximately 0–60 seconds. This occurs because LAT hosts broadcast the LAT services they offer periodically, and the Annex does not update its services table until receiving this broadcast.

Telnet-to-LAT Gateway

The Telnet-to-LAT gateway feature allows the system administrator to associate a unique Internet (IP) address with a specific LAT service. This allows a user to **telnet** to that unique Internet address thereby connecting a user to the associated LAT service.

Before configuring this feature, configure your Annex as described in *Configuring LAT Services* in this section. LAT is operating properly on your Annex if you see your LAT network services appear when you execute the **services** CLI command and you can connect to them using the **connect** CLI command.

To set up the Telnet-to-LAT gateway, the system administrator must add a new entry to the **gateways** file. This new entry has the following syntax:

annex *<ip addr>*
translate telnet <ip addr> to lat <service> [<host> [<port>]]
end

The *<ip addr>* on the *annex-selector* line refers to the specific Annex offering the *gated* LAT service. The *<ip addr>* on the *translate* line is the Internet address that translates to a LAT service. Both *<ip addr>* fields are in the standard decimal dot notation.

The *<service>* field is the desired LAT service name to which **telnet** connects; the *<node>* field is the node that advertises the LAT service; and the *<port>* field is the port name on the LAT node providing the LAT service. The *<host>* and *<port>* fields are optional. The *<service>*, *<host>*, and *<port>* fields can be a maximum of 16 characters. Any errors in the syntax are reported in the syslog file.

Note: There ***must*** be an *annex-selector* line for each set of translate line(s) and the *<ip addr>* on the *translate* line ***must be unique*** on the network and may appear ***only once*** in the **gateways** files. Violating this rule ***will*** cause your network to go down or operate erratically (violation is analogous to defining multiple IP hosts with the same IP address).

Each time a translation entry in the **gateways** file changes, the Annex specified in the *annex-selector* line must be rebooted. In the following example, the Annex with IP address 192.9.200.245 must be rebooted.

```
annex 192.9.200.245
    translate telnet 192.9.200.100 to lat modems node-a
    translate telnet 192.9.200.101 to lat modems
    translate telnet 192.9.200.102 to lat accting
end
```

The above translations are defined only for the Annex with IP address 192.9.200.245. In the first translation, the IP address 192.9.200.100 connects to the LAT service **modems** on the host *node-a*. In the second translation, the IP address 192.9.200.101 connects to the LAT service **modems** on the host reporting the largest rating for **modems**. In the third translation, the IP address 192.9.200.102 connects to a LAT service called **accting** on the host reporting the largest rating for **accting**.

Data-b Slot Support for LAT

The Annex LAT implementation now reports and responds to data-b slot messages.

When a connection is established with a LAT host, the Annex sends that host a report of the port parameters via a report data-b slot message. If the Annex receives a set data-b slot message from a connected LAT host, it responds by configuring the port as commanded by the set data-b slot message.

Parameters changed via data-b slot messages are: parity, baud rate, bits per character, and inband flow control.

Status via data-b slot messages supports the break signal at the local port. To get the Annex to forward this status to the host, disable the local break interpretation.

Miscellaneous LAT Parameters

The LAT-specific **na** parameters, **server_name**, **facility_num**, **service_limit**, **keep_alive_timer**, **circuit_timer**, **retrans_limit** can be changed, but are not necessary to access LAT services; only the **group_value** parameter is necessary for such access. Since the timer and limit values affect performance, take care when adjusting them. For convenience, the administrator may want to change the **server_name** since this is the name by which other LAT hosts will refer to the Annex. After changing the appropriate LAT parameters, the administrator must issue the **na** command **reset annex lat** to activate the new parameters. (*Book D: Reference* describes the **na** and CLI commands in detail.)

Chapter 1

Network Administration

General

This chapter discusses typical software configuration procedures as well as network administration using Annex tools and utilities. Using the Annex, you can:

- Monitor network activity
- Monitor Annex activity
- Secure the network
- Manage the Annex's host table

Monitoring Network Activity

The Annex provides three CLI commands (**arp, ping,** and **netstat**) to monitor network activity (see *Book D: Reference, Chapter 3: CLI Commands* for more details on these commands). Using the CLI commands, you can:

- Display network statistics.
- Test the network.
- Manage the ARP table.

Displaying Network Statistics

The CLI **netstat** command displays statistics and information that the Annex has obtained from the network. You can display hardware statistics for different hardware interfaces.

Hardware Statistics for Ethernet

The following example illustrates interface statistics for an Annex running on an Ethernet LAN:

```
annex01: netstat -i
Name  Mtu   Network     Address   Ipkts Ierrs Opkts Oerrs Collis
en0   1500  132.245.6   annex01   8088  0     3841  0     0
lo0   1536  127         127.0.0.1 0     0     0     0     0
sl16  1006  132.245.6   annex01   14770 0     7468  0     0
sl22  1006  132.245.6   annex01   0     0     0     0     0

                *** Hardware Interface Statistics ***

Ethernet Address:        00-80-2d-00-00-9b
Frames Received:         398681   Frames Transmitted:    452397
Bytes Received:          33965470 Bytes Transmitted:     29453350
CRC Errors:              2        Alignment Errors:      10
Bad Type/Length Fields:  6        Buffer Drops:          0
FIFO Drops:              1        Interface Resets:      1
TX DMA Underruns:        241      RX DMA Overruns:       0
Carrier Sense Losses:    451      Clear to Send Losses:  0
Collisions Detected:     17526    Max Collision Retries:125
```

In the above example, the values do not indicate any problems. Many networks will show proportionally smaller values for the *Bad Type/Length Fields*. Several hosts on this Ethernet do not use the Internet protocol suite; they use a protocol suite that broadcasts packets. The Annex's Ethernet interface receives the packets, but discards them, because it does not recognize the value in the *Type* field.

The *hardware interface statistics* for Ethernet include:

Frames Received	The number of frames received from the network interface.
Frames Transmitted	The number of frames transmitted on the network interface.
Bytes Received	The number of bytes received from the network interface.
Bytes Transmitted	The number of bytes transmitted on the network interface.
CRC Errors	The number of frames received from the network interface with bad a bad CRC.
Alignment Errors	The number of frames received from the network interface that were both misaligned and have a CRC error.
Bad Type/ Length Fields	The number of frames received from the network interface that have an unrecognized type field (ethernet) or an illegal length field (802.3).
Buffer Drops	The number of frames received from the network interface that were good, but dropped because no buffers were available.

FIFO Drops	The number of frames received from the network interface that were lost since the local system bus was not available.
Interface Resets	The number of times the network interface has been initialized from reset; typically, one.
TX DMA Underruns	The number of times a frame transmission is terminated due to lack of data.
RX DMA Overruns	The number of times a frame reception is terminated due to lack of system bus bandwidth.
Carrier Sense Losses	The number of times a frame transmission is terminated due to loss of the Carrier Sense signal. The transceiver cable may have a short or an open.
Collisions Detected	The number of times a frame transmission is terminated due to a collision.
Max Collision Retries	The number times consecutive collisions for a frame exceed the maximum collision retry limit.

Hardware Statistics for a Token Ring Network

The **netstat -i** command displays the following information when executed at an Annex running the IEEE 802.5/Token Ring protocol on a Token Ring network.

```
        *** Hardware Interface Statistics ***
              Adapter Status: Inserted
              Ring Status: None

  IEEE 802.5 Address:  00-80-2d-00-02-a0
  Frames Received:     550234     Frames Transmitted:    433927
  Bytes Received:      61613113   Bytes Transmitted:     2436489
  Data/Signal Errors:  3          Upstream I/F Errors:   0
  Lost Frames:         2          Receive Congestion:    1
  Token Errors:        36         Signal Loss Errors:    4
  Lobe Wire Faults:    0          Ring Recoveries:       12
  Removes Received:    0          Auto-Removal Errors:   0
  Interface Resets:    0          Ring Insertion Errors: 0

        *** IEEE 802.2 Data Link Layer Statistics ***

 802.2 packets received: 433922   Unknown 802.2 types:    0
 Unknown 802.2 SAP's:    0        Unknown SNAP org codes: 0
 Unknown SNAP types:     0
```

The *Adapter Status* indicates whether the adapter is *Inserted* into or *De-inserted* from the ring. A *De-inserted* status indicates that there is a problem with either the network or the network interface card. The network interface driver continuously requests that the adapter try to insert into the ring.

The *Ring Status* indicates conditions or errors existing in the ring. A status of *None* indicates that there are no errors and the adapter and/or the ring is operating normally. If the adapter status is *De-inserted*, the information displayed under *Ring Status* generally provides information on the problem. If most of these errors or conditions occur during the insertion process, the adapter cannot successfully insert into the ring until the problem is corrected. Possible *Ring Status* values include:

AURM Internal adapter error detected during auto-removal MAC frame processing.

The adapter detected a hardware error during the Beacon Auto-removal process. This error causes the adapter to de-insert from the ring.

DUAD Duplicate host address detected during ring insertion phase.

LBWF Lobe wire fault.

The lobe cable is not plugged into the adapter or wiring concentrator, or there is a short in the lobe cable. This error causes the adapter to de-insert from the ring.

RGBE Ring is beaconing.

The source of the problem can be one or more of the following: fault in an adapter on the ring, short or break in the ring cable, or a fault in this station's network adapter.

RGFL Ring failure during ring insertion phase.

The source of the problem can be one or more of the following: fault in an adapter on the ring, short or break in the ring cable, or a fault in this station's network adapter.

RGRE Ring conducting recovery action.

RMRC Remove MAC frame received.

The network manager has forced the adapter off the ring. This error causes the adapter to de-insert from the ring.

RQIN Request ring parameter fault detected during ring insertion phase.

The source of the problem can be one or more of the following: fault in an adapter on the ring, or a fault in this station's network adapter.

SGLO Signal loss detected.

The source of the problem can be one or more of the following: fault in the upstream adapter on the ring, short or break in the ring cable, or a fault in this station's network adapter.

SLST The station detected that it is the only station on the ring.

TIMO Time-out during ring insertion phase.

XMBE This adapter is transmitting beacon MAC frames.

The source of the problem can be one or more of the following: fault in the upstream adapter on the ring, short or break in the ring cable, or a fault in this station's network adapter.

Other *hardware interface statistics* for a Token Ring network include:

Frames Received	The number of frames received from the network interface.
Frames Transmitted	The number of frames transmitted to the network.
Bytes Received	The number of bytes received from the network interface.
Bytes Transmitted	The number of bytes transmitted to the network.
Data/Signal Errors	The number of times a possible line hit, corrupted frame control field, or frame check sequence error was detected.
Upstream I/F Errors	The number of times an upstream adapter was unable to set its Address Recognized Indicator bit and/or its Frame Copied Indicator bit.
Lost Frames	The number of times an end-of-frame delimiter was not detected during frame reception.
Receive Congestion	The number of times the adapter recognized a frame for itself but has no receive buffer space.
Token Errors	The number of errors in the token or the number of times the token was lost.
Signal Loss Errors	The number of times the adapter has detected a signal loss on the ring.
Lobe Wire Faults	The number of times the adapter detected a short in the lobe cable.
Ring Recoveries	The number of times the ring entered one of three ring recovery modes.
Removes Received	The number of times the adapter was forced off the ring by the network manager via a Remove Station MAC fame.
Auto-removal Errors	The number of time the adapter detected a hardware error during the Beacon Auto-removal Process.
Interface Resets	The number of times the driver had to re-initialize the token ring card due to an unrecoverable exception in the card.
Ring Insertion Errors	The number of times the network interface driver attempted to insert the adapter into the ring and failed.

The *IEEE 802.2 data link layer statistics* include:

802.2 packet received	The number of frames received with a recognizable 802.2 header.
Unknown 802.2 SAP's	The number of unknown 802.2 Service Access Points received.
Unknown 802.2 SNAP's	The number of unknown 802.2 Subnetwork Access Points received.
Unknown 802.2 types	The number of 802.2 headers specified with an unknown IP type received.
Unknown SNAP org code	The number of 802.2 headers that specified an unknown organization code in the SNAP portion of the header.

SLIP Statistics

The **netstat -iS** command displays additional SLIP receive and transmit data at the end of the hardware interface statistics display. For example:

```
annex01 v1# netstat -iS
Name    Mtu     Network     Address     Ipkt    Ierrs   Opkts   Oerrs   Collis
en0     1500    192.9.200   annex1      648918  0       352845  0       0
lo0     1536    127         127.0.0.1   0       0       0       0       0
sl6     1006    192.9.200   annex1      0       0       0       0       0
cslip   256     192.9.200   annex1      0       0       13      0       1

                *** Hardware Interface Statistics ***

 Ethernet Address:       00-80-2d-00-14-3d
 Frames Received:        705482          Frames Transmitted:    352839
 Bytes Received:         62425605        Bytes Transmitted:   19357865
 CRC Errors:             0               Alignment Errors:          0
 Bad Type/Length Fields: 0               Buffer Drops:              0
 FIFO Drops:             0               Interface Resets:          1
 TX DMA Underruns:       0               RX DMA Overruns:           0
 Carrier Sense Losses:   0               Clear to Send Losses:      0
 Collisions Detected:    2389            Max Collision Retries:     0

 SLIP rcvr:
      intrs 0, loops 0, bytes 0, pkts 0
      bytes/intr 0, bytes/loop 0, bytes/pkt 0
      hiwaters 0, overflows 0, mbuf waits 0, mbuf kicks 0
      overruns 0, ipintrq full 0
      FRAME_ENDs 0, FRAME_ESCs 0, proto errs 0, last proto err 0
 SLIP xmit:
      intrs 13, starts 22, vectors 108, bytes 1874, pkts 13
      FRAME_ENDs 22, FRAME_ESCs 32
      bytes/intr 144, bytes/vec 17, vec/pkt 8, bytes/pkt 144
```

Active Connections

If you enter the **netstat** command without options, it displays the local and remote addresses, send and receive queue sizes (in bytes), protocol, and the internal state of the protocol for all active connections. The **-A** option adds the protocol control block (PCB) addresses. The **-a** option includes sockets used by server processes and can be used in combination with the **-A** option. For example:

```
annex01 v1# netstat -a
Active connections (including servers)
Proto Recv-Q Send-Q Local Address      Foreign Address        (state)
tcp        0      2 annex1.telnet     test1.4759             ESTABLISHED
tcp        0      0 annex1.883        gibbs.login            ESTABLISHED
tcp        0      0 annex1.1085       nepal.telnet           ESTABLISHED
tcp        0      0 annex1.1082       nepal.telnet           ESTABLISHED
tcp        0      0 annex1.1081       opus.telnet            ESTABLISHED
tcp        0      0 annex1.1022       test1.login            ESTABLISHED
tcp        0      0 annex1.703        test1.login            ESTABLISHED
tcp        0      0 annex1.797        test1.login            ESTABLISHED
tcp        0      0 annex1.707        test1.login            ESTABLISHED
tcp        0      0 annex1.957        test1.login            ESTABLISHED
tcp      211      0 annex1.953        xxzzyx.login           ESTABLISHED
tcp        0      0 annex1.1021       test1.login            ESTABLISHED
tcp        0      0 *.finger          *.*                    LISTEN
tcp        0      0 *.printer         *.*                    LISTEN
udp        0      0 *.snmp            *.*
udp        0      0 *.513             *.*
udp        0      0 *.erpc            *.*
udp        0      0 *.route           *.*
udp        0      0 *.1024            *.*
```

Routing Table Information

The **netstat -r** command displays statistics and information about all available routes in the routing table. A route comprises a destination host or network and the gateway through which data is forwarded. For example:

```
annex01 v1# netstat -r
Routing tables
Destination      Gateway            Flags    Refcnt Use        Interface
127.0.0.1        127.0.0.1          UHF      0      0          lo0
192.9.200.130    annex1             UHF      0      0          sl6
annex01          127.0.0.1          UHF      0      0          lo0
192.9.200.175    annex1             UHF      1      9          cslip10
192.9.200.0      annex1             UF       16     351713     en0
192.9.201.0      192.9.200.101      UGF      0      0          en0
192.9.202.0      mole               UG       0      0          en0
192.9.100.0      192.9.200.223      UGF      0      0          en0
132.245.0.0      test1              UG       0      0          en0
69.0.0.0         oliver             UGF      0      0          en0
```

The *Refcnt* field displays the number of current users of the route. The *Use* field displays the number of packets sent using that route. The *Interface* field displays the network interface using the route. The *Flags* field displays:

- U for up
- G for a gateway
- H for a route to a host
- D for a route added because of an ICMP redirect
- F for routes defined as hardwired
- P for gateways defined as passive

Rotary Information

The **netstat -R** command displays all rotaries configured for the Annex. The *Rotary name* displays the name of the rotary. The *Address* field displays the auxiliary address, if assigned, or an asterisk (*), indicating the rotary has the same address as the server. This field also displays the configured TCP port as either **.telnet** or a number in the 6000 range. The *Proto* field displays the assigned protocol. The *Camp* field displays the camp-on options: **ask, always,** or **never**. The *Flags* field displays **I** if the rotary is invisible. The *Annex ports* field displays the port(s). For example:

```
nepal v1#   netstat -R
Rotary name       Address                 Proto   Camp  Flags  Annex ports
oemandy1          *.telnet                telnet  ask          11
conan_33          *.telnet                telnet  ask          16
borneo1           192.9.200.250.telnet    telnet  ask          1
brazil7           192.9.200.253.6003      telnet  ask          7
annex3            *.6103                  telnet  ask          8,13,15
```

Memory Statistics

The **netstat -m** command displays statistics for the memory management routines. For example:

```
annex01 v1# netstat -m
164/599 mbufs in use:
        14 mbufs allocated to data
        12 mbufs allocated to packet headers
        19 mbufs allocated to socket structures
        33 mbufs allocated to protocol control blocks
        10 mbufs allocated to routing table entries
        8  mbufs allocated to socket name
        4  mbufs allocated to interface address
        64 mbufs allocated to incoming network i/f packets
149 Kbytes allocated to network (27% in use)
0 requests for memory denied
```

Protocol Statistics

The **netstat -s** command displays statistics for each of the five protocols: ICMP, UDP, TCP, IP, and LAT. The LAT statistics display only if the correct **lat_key** value has been set. For example (a truncated view):

```
annex01 v1# netstat -s
tcp:
        119153 data packets sent
        348537 packets sent
                63 data packets (1888 bytes) retransmitted
                :
                533421 packets received
                :
                7 connections dropped by keepalive
udp:
        0 incomplete headers
        0 bad data length fields
        0 bad checksums
        2755 no listening port
        77148 packets received
        956 packets sent
ip:
        613422 total packets received
        0 bad header checksums
        :
        4 output packets we did frag
        5 output fragments we created
icmp:
        2359 calls to icmp_error
        0 errors not generated 'cuz old message too short
        0 errors not generated 'cuz old message was icmp
        Output histogram:
                destination unreachable: 2358
                :
        Input histogram:
                echo reply: 41
                :
lat:
        241 Total run messages received
        228 Total run messages transmit
        :
        56382 Total service messages recv.
        3796 Total service messages used
```

Testing the Network

The superuser CLI **ping** command tests and measures the LAN. Additionally, it can isolate a single-point hardware or software failure. The **ping** command sends out an Internet Control Message Protocol (ICMP) echo request packet each second, or until input from the terminal terminates the command. After completing, **ping** displays a summary of all echo replies received. This display includes a calculation of the time, in milliseconds, that it takes to return the message. Following is an example of the display:

```
annex01# ping caddy
PING caddy: 56 data bytes
64 bytes from 132.245.6.25: icmp_seq=0. time=37. ms
64 bytes from 132.245.6.25: icmp_seq=1. time=12. ms
64 bytes from 132.245.6.25: icmp_seq=2. time=12. ms
64 bytes from 132.245.6.25: icmp_seq=3. time=12. ms
----caddy PING Statistics----
4 packets transmitted, 4 packets received, 0% packet loss round-trip (ms)
min/avg/max = 12/20/37
```

When using **ping** for troubleshooting, **ping** the Annex first. For example:

```
annex# ping annex
```

After verifying that the Annex's hardware interface is operating properly, **ping** individual hosts that are distanced from the Annex: first **ping** the host that is physically closest to the Annex, then **ping** a host that is a little farther away.

Managing the ARP Table

The Address Resolution Protocol (ARP) maps Internet addresses to hardware addresses. Hosts implementing ARP maintain a translation table for these address mappings. When an Annex receives a request for a host that does not have a translation entry in the ARP table, it broadcasts for the hardware address. The superuser CLI **arp** command displays and modifies entries in this translation table.

Since the Annex automatically builds the ARP table dynamically, you rarely need to modify the table. You can use **arp** to modify the table for hosts that do not implement ARP, enabling communications between the host and the Annex. Using **arp**, you can:

- Delete a specified entry
- Create an entry for a host

A created entry is permanent unless it is defined as temporary, in which case the entry is deleted after 20 minutes. An entry defined as published causes the Annex to respond with its hardware address for the specified host, even though the Internet address is not the Annex's. Publishing a hardware address for another host frequently is done to route data to a host connected to the Annex through a SLIP link.

Monitoring Annex Activity

Monitoring Annex activity involves:

- Logging user and Annex activities
- Displaying user activity
- Displaying Annex statistics
- Monitoring serial line activity

The Annex provides CLI commands to assist in these tasks. Refer to *Book D: Reference, Chapter 3: CLI Commands*, for more details on using these commands.

Logging User and Annex Events

The Annex provides two mechanisms for logging events: host-based security and the 4.3BSD system logging daemon.

Host-based Security Logging

Host-based security provides logging capabilities that maintain audit trails of user activity. The security server logs the occurrence of the following events:

- Annex boots
- User login activities in response to granting access to a CLI
- User logout from CLI
- Rejections for access to CLI
- User login activities to the port server
- Timeouts on a port

The security server logs each event as a message to its file **acp_logfile**. Security logging is enabled automatically when host-based security is enabled for an Annex (using the **security_enable** Annex parameter).

Events are logged to the security server that responded to the security request, either granting or denying access requests. When using a back-up security server(s), the **acp_logfile** is located on all servers. Each logged message contains the following fields:

- The client Annex's Internet address
- Sequence number
- Port number
- Date
- Time
- Protocol
- Event
- Protocol-dependent information

Immediately after downloading its image, the Annex randomly generates the sequence number, and initializes it as four random hexadecimal digits, followed by four zeroes. The sequence number increments with each security request. The date and time are expressed as *yymmdd* and *hhmmss* (24-hour time). The protocols are:

- **security** for non-specific events (e.g., boot and security reset messages)
- **cli** for CLI access
- **pserv** for port server access
- **rlogin, telnet,** and **connect** for interactive connections

All fields are separated by colons and are encoded for use by UNIX utilities that sort, merge, select, or filter streams. When more than one host functions as a security server, the log files can be merged and sorted by the date and time fields. Following is a sample default log file (the format can be changed):

```
May 1  00:04:43  rosco  timed[579]:adjusting time from host 192.9.200.95:old = 281e3e5a...delta = 1
May 1  00:17:12  milo   server[4214]:Port-Exit:14:RDP:Qume ScripTEN #0:Engineering:xenna
May 1  00:37:00  oliver cli[4446]:Job-Begin:23:rlogin steam:tom:Dial-in 800
May 1  00:43:42  oliver cli[4446]:Job-Exit:23:rlogin steam:tom:Dial-in 800:Connection closed
May 1  01:29:16  radar  root[0]:route:add dst 69.0.0.0 via gateway 192.9.200.201
May 1  01:29:17  radar  root[0]:route:add dst 192.9.201.0 via gateway 192.9.200.101
May 1  01:49:09  oliver cli[4547]:Job-Begin:22:telnet:gimme:tb + D1
May 1  01:49:09  oliver telnet_cmd[4671]:Telnet-Begin:22:telnet xylogics.com uucp
May 1  01:49:49  oliver cli[4547]:Job-Exit:22:telnet xylogics.com uucp:gimme:tb + D1:Cnctn closed
May 1  05:53:05  oliver slip[4668]:Port-Exit:24:SLIP:slip to:mfg network:oliver
May 1  06:17:11  wally  telnet_cmd[3955]:Port-Begin:21:DP:hallo:V2-5/M2-1-1:wpvax
May 1  08:04:38  frodo  cli[800]:Port-Begin:3:CLI:swm:eng x:[local]
May 1  08:04:41  frodo  cli[800]:Job-Begin:3:rlogin xenna:swm:eng x
May 1  09:04:13  spiff  cli[374]:Job-Begin:18:connect wpvax:wdt:is-1,tb2500
May 1  09:04:27  spiff  cli[374]:Job-Exit:18:connect wpvax:wdt:is-1,tb2500:Error 0
```

Events are continuously written to the file **acp_logfile**. To prevent this file from overwhelming the file system on the hosts, and still obtain the record information for generating reports, move the file at regular intervals. The size of your network, the number of Annexes, and the amount of activity generated at each Annex determines the frequency for moving the file. Consider starting on a weekly basis.

Event Logging Using syslog

The Annex can log events for a system running a 4.3BSD-style **syslog** daemon. The logged message includes:

- The date and time of the event
- The name or Internet address of the Annex on which the event occurred
- The name of the event and PID of the Annex process
- A description of the event

In the following example, on May 5, at 9:19 a.m., a user named *Worth* on port 8 of *annex01* issued the **rlogin** command to host *galago*.

```
May 5 9:19:03 annex01 cli[598]:Job-Begin:8:rlogin galago:Worth
```

The information display differs, depending on the event. In the following example of a typical message, a time server updates the Annex's time. The time server host's address displays in hexadecimal longword. Times are expressed in hexadecimal as the number of seconds since 00:00:00 January 1, 1970.

```
Jan 5 9:56:5 annex timed[38]:adjusting time from host
      5fc809c0: old=25bf1398, new=25bf1399, delta=1
```

The next example shows a user on port 9 of *annex* issuing the **telnet** command to access another Annex.

```
May 5 8:56:3 annex telnet_cmd[35]:Telnet-Begin:9:telnet annex1
```

The next example shows a request for the printer on *annex* through the port server.

```
May 5 8:17:5 annex rdr[39]:Port-Begin:14:RDP:LPRt10:Actg:ager
```

You can create audit trails and accounting reports for the Annex and its serial ports by sorting and merging these logs.

Displaying User Activity

The CLI **who** command displays information on the current users of the Annex's ports. This command also displays current users on other Annexes, and on other hosts, if those hosts run a **who** daemon, such as **whod**, distributed with the 4.3BSD system. When the **who** command is issued for an Annex, it displays the user name, the jobs the user is running, when the connection began, any idle time, and the connection's origin for each serial port and virtual CLI connection. For example:

```
annex01: who
 Port What User     Location   When     Idle  Address
  2   CLI  sally    Ext 528   8:50am          [local]
      +1 'rlogin borneo'
  8   CLI  larry    Ext 638  Mon 9:33am 9:06  [local]
      +1 'rlogin botswana'
      -2 'telnet thirdfloor'
 12   CLI  jane     Ext 64   May 5 9:44 15:58 [local]
 15   PSVR  ---     lqp port  10:15am          borneo
 16   CLI  barry    Ext 627    9:23am         [local]
      +1 'telnet borneo'
 v1   CLI   kathryn    ---     11:20am           thyme
```

When the command is issued for a 4.3BSD host, the display is the same as for the **finger** command executed at the host. Using the **who** command, you can obtain a significant amount of information on users and their activities in the network. Using this command, you can display:

- All users connecting from or to a specific host(s)
- A single user or a group of users connected to the Annex
- All users connected to specific port or virtual CLI
- A specific user or all users logged into a specified host

You can display a range of hosts or user names by abbreviating them.

Displaying Annex Statistics

The CLI **stats** command displays general Annex statistics, or statistics for one or more serial ports. The following example shows part of a typical display for an Annex on an Ethernet network.

```
S/W Version:ANNEX3-UX R6.0.1  Build #1:Fri Mar 15 19:23:35 EST 1991
ROM Rev:0505.  H/W Type:Annex3 32.  H/W Rev:1.2.  Serial Number:1.
Uptime:22 hours 42 mins          Date:Wed May  1 10:19:12 1991 EDT
Image booted:oper.42.enet
Booted from:192.9.200.95
Subnet mask:255.255.255.0        Broadcast addr:192.9.200.255
Inet addr:192.9.200.228          Ethernet addr:0-80-2d-0-14-20
Default domain:i.xylogics.com
Loading:
    CPU = 5%  procs active/max/limit=41/42/320
    rescheds=0/22  switches=23/113526  activates=23/113556
Callouts:
    max=227  free=212  minimum free=210
Cblocks:
    total=55552  free=55552  minimum free=55552  denied=0
Serial Ports:
    Total bytes:rcv'd=11980  xmt'd=1117219
    Errors:parity=0  framing=0  fifo overruns=0
Memory:
    total=2097152 available=1413376 free=770648 minimum free=767248
```

Using the **stats** command, you can display statistics for a specific serial port, a range of serial ports, or all serial ports. The following example shows the format:

```
annex01: stats -s
P#  Control Lines   Speed     CharTx    CharRx    Parity   Overrun  Framing
 1      none        38400      255        0         0        0        0

 2      CTS RTS      4800      255        0         0        0        0

 3      none        19200      255        0         0        0        0

 4      DTR DCD DSR 38400      176715     4123      0        0        0

 5      DTR DCD DSR  9600      937802     7864      0        0        0

 6      idle        idle       0          0         0        0        0
 :
 :
total                          1118837    11987     0        0        0
```

Monitoring Serial Line Activity

The Annex provides two superuser CLI commands that display information about the state of the Annex's serial ports: **control** and **tap**. *Book D: Reference, Chapter 3: CLI Commands*, provides more detail on these commands.

The superuser CLI **control** command is a diagnostic tool that, for a specified port, allows you to set DTR and RTS, monitor DCD, DSR, and CTS, or output a short test message. The superuser CLI **tap** command accesses (wiretaps) a serial port from a terminal. Using **tap**, you can:

- Observe the output to the port. The command also displays keystrokes entered from your terminal as output to the port you are tapping as if they had been entered on the port.
- Find out exactly what users are seeing on their terminals from a remote location.
- Provide on-line advice and instructions to users at their terminals.
- Monitor traffic in both directions on the port, especially incoming special conditions, such as line breaks, flow control, and special characters.

Under certain circumstances, the order of displayed data may not match the actual time sequence of the events. All input and output data is displayed. Special characters and control line changes are stored in a limited buffer. If these changes occur too rapidly, they may be lost.

Securing the Network

The Annex security system provides a full set of features that allow you to establish any security policy required by the network. You can choose between host-based and local password protection on the Annex, or you can use local password protection as a back-up for host-based security.

The Annex Administrative Password

The Annex administrative password protects the administrative tools. Each Annex comes with an assigned administrative password: its Internet address. When the **na** command **show annex** displays the password as "**<unset>**", use the default administrative password for:

- Access to superuser CLI commands.
- Access to ports locked with the CLI **lock** command.
- Access to a virtual CLI connection through local password protection.

Modifying the assigned administrative password enables:

- Password protection on access to Annexes through **na**.
- Encryption of messages between an Annex and **na**.

Note: If you lose your administrative password, your only option is to erase the Annex's non-volatile memory using the ROM Monitor **erase** command, and re-enter all parameters.

Protecting Ports from Unauthorized Access

When terminals are connected to a network, they provide users with the potential for unauthorized access to hosts and resources. In addition to the available security schemes, the Annex provides timers that can reset a port. The **cli_inactivity** port parameter sets the CLI inactivity timer: when the last session is completed, the Annex resets the port after the amount of time specified in this parameter has elapsed.

Users can protect their login sessions using the CLI **lock** command if they do not want to log out when leaving the terminal unattended.

Protecting the Superuser CLI

An Annex administrative password is required for access to the superuser CLI using the **su** command. The default password is the Annex's Internet address. The password can be changed in two ways:

- Using the superuser CLI **passwd** command.
- Changing the **password** parameter using **na**.

Using either method, the new password takes effect immediately for access to the superuser CLI.

Reset the password to the Annex's Internet address by:

- Using **na** to set the **password** parameter to the null string. For example:

```
command: set annex password ""
```

- Using the superuser CLI **passwd** command and pressing **Return** in response to the prompt for a new password.
- Erasing all parameters using the ROM monitor **erase** command.

Preventing Unauthorized Access to na

When using **na**, users can access Annex parameters and obtain useful information, or reconfigure and reboot Annexes. Protecting **na** involves UNIX superuser protection and the Annex administrative password.

The installation procedure installs **na** as owned by root and executable by all. Only a superuser can execute the **set, reset, broadcast, dumpboot, boot, read,** and **copy** commands.

Managing the Host Table

The host table contains this information for each host:

- Host name
- Aliases (if any)
- Internet address
- Multiple Internet addresses (if any)
- System status (if the entry is built from RWHO)
- Load factor (if the entry is built from RWHO)
- Number of users (if the entry is built from RWHO)

The CLI **hosts** command displays all entries in the host table. The Annex can build and update the host table from RWHO messages, and from responses to DNS and/or IEN-116 queries. Entries are updated according to information received. Information for a host will be updated if new information received is different from what is currently in the host table. The Annex considers information from a DNS server the most reliable source; it considers an IEN-116 as the next reliable source; and it considers RWHO broadcasts as the least reliable source. Thus, information from a DNS server always updates current information received from either an IEN-116 server or an RWHO broadcast; information from an IEN-116 server always updates current information received from an RWHO broadcast.

The Annex also deletes entries. The criteria for deletion depends on the source of the entry. Each DNS response includes a time to live (TTL). When an entry reaches its full life (default = 60 minutes), the DNS server is queried again. If a DNS server recognizes the name, the entry is re-entered in the host table; otherwise, it is deleted. The Annex keeps track of how often each IEN-116 host table entry is referenced. If a name server entry has not been used for 32 days, it is deleted.

The Annex expects to receive an RWHO message from a host at least every six minutes; if no message is received in that time period, the host table status entry for that host is changed to *down ?*. If there is no message for 12 minutes, the status is changed to *down*, and if no message is received for 60 minutes, the entry is removed from the table.

If the host table acquires a new entry after it is full, the Annex deletes the oldest, least-used entry to make room for the new one. If the host table is too small, it frequently changes. Increasing the size of the host table using the **host_table_size** Annex parameter reduces these changes.

Two other tools for managing the host table are the CLI superuser **host -f** command and the **na** command **reset annex nameserver**. The **host -f** command flushes all, or specified, entries in the host table. The **reset** command resets all name server parameters discussed in this section and flushes all entries from the host table.

Additionally, the **gateways** file permits a line entry containing a host name that is associated with an Internet Address. This entry is similar to the **/etc/hosts** file entry, except aliasing is not supported. When the Annex boots, it adds this host name entry to the host table. These table entries live in the host table until a nameserver overrides the entry's information or until the administrator resets the Annex nameserver via the **na** or CLI **admin** commands. For more information, see *Book B: Configuration Procedures, Chapter 7: Configuring Hosts and Servers, Host Table Initial Loading in the gateways File*.

Typical Configuration Problems

Each Annex hardware platform provides a hardware installation guide that contains troubleshooting information. Many problems that occur after an Annex is running are due to improper configuration of the Annex or a host. The following subsections describe the symptoms of several common configuration problems.

Sessions not Terminated

Several situations can leave a session open. On CLI ports, the **hangup** command may not disconnect a modem or a switch. On CLI login ports, a modem, telephone, or switch disconnection (de-asserting DCD) may not terminate the CLI connection or UNIX session: the next port user finds a CLI connection with jobs already active and does not receive a security prompt, or receives a shell prompt without logging in.

Another symptom of this problem is that a port configured as autobaud retains the baud rate of the previous session. The port server session may not be terminated if you try to use an outgoing Annex port as a front-end to another host (or to connect to a modem or switch), and the interface at the other end drops DCD. If this occurs, configure the modem port's parameters correctly (see *Book B: Configuration Procedures, Chapter 5: Modems*). Configure this port as **type dial_in** and **control_lines modem_control**. Check the cable wiring, and pay close attention to the wiring of the Annex's DCD, DSR, and DTR control lines. The superuser CLI **tap** and **control** commands can provide useful information. When changing parameters using **na**, remember to use the **reset** command after entering the new values.

Connection Delays When Using Name Servers

Annex users may notice connection delays under certain circumstances. If **name_server_1** and **name_server_2** are defined, and **name_server_1** is down or does not exist, there will be a 15–30 second delay until **name_server_2** resolves the name during a connect to a host using **rlogin** or **telnet**. If both name servers are down or they do not exist, there will be up to a 45 second delay. If the host to which the user ID is trying to connect is in not in the RWHO host table, an error occurs; the terminal displays a message informing the user that the name server is unreachable.

Hosts not Appearing in Hosts Display

The Annex **hosts** command should list any hosts that broadcast RWHO packets if the **na** parameter **rwhod** is set to **Y**. If you expect to see a host in the **hosts** display and it does not appear, wait several minutes and then re-issue the **hosts** command before assuming there is a problem; the time between broadcasts can vary. Before proceeding, verify that the host not appearing in the **hosts** display is sending RWHO packets correctly by entering **ruptime** on another host on the network, or by checking that the host in question is running **rwhod**.

If the host is sending RWHO packets correctly, incompatible broadcast addresses may be causing the problem. Originally, a broadcast packet used a host address of all *zeros* (*network.0*) Later refinements required a change to the broadcast address, specifying a host address of all *ones* (*network.1*). A host configured with a *network.1* address will accept *network.0* broadcasts. Hosts configured with *network.0* addressing will not see *network.1* broadcasts.

You can configure the Annex to use either method of addressing by setting the **broadcast_addr** parameter.

Wrong Host Address in Host Table

The Annex assumes that the host described in the data part of the RWHO packet sent the packet, and the IP header's *source-Internet-address* field contains the host's address. Usually, this assumption is correct because routers do not forward broadcast packets. Some RWHO daemons do forward RWHO packets.

When a router forwards RWHO packets, the IP header contains the address of the forwarding host. The Annex then associates the host described in the packet with the wrong address. In some cases, you can change the behavior of the RWHO daemon on hosts routing the packets. If you cannot alter this behavior and you find it a problem, you can turn off RWHO at the Annex by setting the **rwho** parameter to **N**. RWHO entries are not added to the Annex's host table.

Network Logins to BSD Hosts are Invisible

An Annex user can **rlogin** or **telnet** to a host, but the pseudo-terminal does not show up in a **who** command display. This problem is caused by a mismatch between pseudo-terminals configured in the **/dev** directory and pseudo-terminal entries in **/etc/ttys**. Update the **/etc/ttys** file to contain the proper number of pseudo-terminals as indicated by the actual device entries in **/dev**.

All Network Ports are in Use

The **rlogin** or **telnet** command is rejected after the user name is entered in response to the **login:** prompt. The error message *all network ports in use* indicates that all available pseudo-terminals are in use. On BSD hosts, update **/etc/ttys** and create more pseudo-terminals in **/dev**.

Chapter 2

Simple Network Management Protocol (SNMP)

General

The Annex provides a Simple Network Management Protocol (SNMP) agent. This agent implements the current standard Management Information Base (MIB), both MIB and MIB-II as defined in RFC1156 and RFC1158 respectively, as well as Annex-specific variables. Entries in the Annex's **gateways** file both enable the SNMP agent and define the operating characteristics of the SNMP daemon that controls the SNMP agent.

Configuring the SNMP Agent

Configure and enable the SNMP agent by adding lines to the Annex **gateways** file. The Annex provides four SNMP commands: **community, traphost, contact,** and **location.**

The **community** command defines the SNMP community name from which the Annex responds to requests. You can specify more than one SNMP community name but each community requires a separate line. To enable the SNMP agent, you must include at least one **community** command in the **gateways** file. The command syntax is:

snmp community *name*

Traps are unsolicited administrative messages generated by SNMP agents on the network. The **traphost** command defines the host to which SNMP traps are sent. For the Annex to generate traps, at least one *trap host* must be configured and the SNMP community **public** must be defined. You can specify more than one trap host, but each host requires a separate line. Specify the trap host using its Internet address (RFC1157 provides more details on communities and traps). The command syntax is:

snmp traphost *inetaddr*

The supported SNMP traps are:

- **coldstart** upon initialization of the SNMP agent at boot
- **link-up** upon initialization of each network interface
- **link-down** upon de-configuration of any network interface

Note: **Link-up** and **link-down** apply only to SLIP interfaces, since they are the only interfaces that can change operational states.

The **contact** command defines the object that identifies the person responsible for managing the Annex, as supported by MIB-II. The command syntax is:

snmp contact *string*

The *string* can include information about how to contact the person; e.g., *John Smith, ext. 370.*

The **location** command defines the object that describes the Annex's location; e.g., *computer room, 2nd floor* or *engineering lab*. The command syntax is:

snmp location *string*

The following example shows an entry in the **gateways** file using the SNMP command:

```
annex 132.245.6.34
    host 132.245.1.01 gateway 132.245.7 metric 1 hardwired
    net 132.245.9.0 gateway 132.245.2.3 metric 1 hardwired
    snmp contact john smith ext 370
    snmp location computer rm 2nd flr
end

snmp community public
snmp traphost 132.245.6.50
```

Annex Private Enterprise MIB

This section describes each of the objects within the Annex private enterprise Management Information Base (MIB). These descriptions follow the format used in RFC1156:

OBJECT	The name of the OBJECT DESCRIPTOR for the object type: includes the OBJECT IDENTIFIER.
Syntax	The abstract syntax for the object type in Abstract Syntax Notation One (ASN.1).
Description	A description of the semantics of the object type. This description ensures that all implementations interpret the object type appropriately and that all object types are consistent across all machines.

RFC1156 requests two additional descriptors: Access and Status. To reduce the amount of space required to list all of the variables, these two descriptors are not included. All variables in the Annex private enterprise MIB are defined as Access (read-only) and Status (mandatory).

Object Definitions

This section provides the object definitions for the Annex private enterprise MIB for the hardware, software, and ports groups. The Annex software distribution provides this information in the file */annex_root*/**src/snmp/annex-mib.txt**. The full object identifier path is:

ANNEX-MIB { iso org(3) dod(6) internet(1) private(4) enterprises(1) 15 }

```
xylogics      OBJECT IDENTIFIER ::= { enterprises 15 }

prod          OBJECT IDENTIFIER ::= { xylogics 1 }
prod_annex    OBJECT-TYPE :: = { prod 1 }

annex         OBJECT IDENTIFIER ::= { xylogics 2 }

hw            OBJECT IDENTIFIER ::= { annex 1 }
sw            OBJECT IDENTIFIER ::= { annex 2 }
ports         OBJECT IDENTIFIER ::= { annex 3 }
```

The full object identifier path for the Annex variable is: 1.3.6.1.4.1.15.2.

The Hardware Group

The following variables describe the specific Annex hardware platform.

OBJECT: hwType { hw 1 }

Syntax: Integer {
err(1), annexII(16), annex3(42), microannex(52)
}

Definition: Hardware platform type.

OBJECT: hwRev { hw 2 }

Syntax: Octet string

Definition: Revision level of the hardware platform.

OBJECT: romRev { hw 3 }

Syntax: Octet string

Definition: Revision number of the ROM Monitor software. Some original Annexes may not return this value.

OBJECT: serialNumber { hw 4 }

Syntax: Integer

Definition: Serial number assigned to the Annex.

OBJECT: memorySize { hw 5 }

Syntax: Integer

Definition: Amount of memory available for the hardware platform.

The Software Group

The following variables describe the software used by the Annex.

OBJECT: swType { sw 1 }

Syntax: Integer {
err(1), annexIImx(16), annexIIux(17),
annex3ux(42), annex3mx(43)
microannexux(52), microannexmx(53)
}

Definition: Software version running in the Annex.

OBJECT: swRevMajor { sw 2 }

Syntax: Integer

Definition: Number of the software release.

OBJECT: swRevMinor { sw 3 }

Syntax: Integer

Definition: Number of the software point release.

OBJECT: swBuild { sw 4 }

Syntax: Octet string

Definition: Full name of the software build, including software version, major and minor version numbers, and date and time of the build.

OBJECT: imageName { sw 5 }

Syntax: Octet string

Definition: Name of the image from which the Annex last booted. Some original Annexes may not return this value.

OBJECT: bootHost { sw 6 }

Syntax: IpAddress

Definition: Internet address of the host from which the Annex booted. Some original Annexes may not return this value.

OBJECT: defaultDomain { sw 7 }

Syntax: Octet string

Definition: Default Domain name provided by the DNS name server.

OBJECT: currentDate { sw 8 }

Syntax: Octet string

Definition: Current date and time provided by the time server.

OBJECT: usableMemory {sw 9 }

Syntax: Integer

Definition: Amount of available memory for this Annex after it has booted.

OBJECT: freeMemory { sw 10 }
Syntax: Gauge
Definition: Amount of available memory.

OBJECT: minFreeMemory { sw 11 }
Syntax: Integer
Definition: Least amount of free memory recorded since the Annex has been up.

OBJECT: cpuUtilization { sw 12 }
Syntax: Gauge
Definition: Percent of CPU time currently being used.

OBJECT: maxProcs { sw 13 }
Syntax: Integer
Definition: Total number of processes allocated for use in the Annex.

OBJECT: mostProcs { sw 14 }
Syntax: Integer
Definition: Maximum number of active processes that have been recorded since the Annex was last booted.

OBJECT: activeProcs { sw 15 }
Syntax: Gauge
Definition: Current number of active processes.

OBJECT: cpuReschedsI { sw 16 }
Syntax: Gauge
Definition: Number of processes whose time slice has expired in the last minute.

OBJECT: cpuReschedsT { sw 17 }
Syntax: Counter
Definition: Total number of processes whose time slice has expired since the Annex was last booted.

OBJECT: contextSwI { sw 18 }

Syntax: Gauge

Definition: Number of context switches in the past minute.

OBJECT: contextSwT { sw 19 }

Syntax: Counter

Definition: Total number of context switches since the Annex was last booted.

OBJECT: cpuActivatesI { sw 20 }

Syntax: Gauge

Definition: Number of processes that were activated in the last minute.

OBJECT: cpuActivatesT { sw 21 }

Syntax: Counter

Definition: Total number of processes that were activated since the Annex was last booted.

OBJECT: cBlocksTotal { sw 22 }

Syntax: Integer

Definition: Total number of available serial line output buffers (Cblocks) in bytes.

OBJECT: cBlocksFree { sw 23 }

Syntax: Integer

Definition: Number of Cblocks in bytes currently free.

OBJECT: cBlocksMinFree { sw 24 }

Syntax: Integer

Definition: Minimum number of Cblocks in bytes that have been available since the Annex was last booted.

OBJECT: cBlocksDenied { sw 25 }

Syntax: Counter

Definition: Total number of times a Cblock request was denied because none were free.

OBJECT: maxCallouts { sw 26 }

Syntax: Integer

Definition: Total number of Callout structures for timed events available in the Annex.

OBJECT: leastCallouts { sw 27 }

Syntax: Integer

Definition: The least number of Callout structures that have been available since the Annex was last booted.

OBJECT: freeCallouts { sw 28 }

Syntax: Gauge

Definition: Current number of free Callout structures.

The Ports Group

The following variables provide details about Annex ports.

OBJECT: totalPorts { ports 1 }

Syntax: Integer

Definition: Total number of serial ports for this hardware platform.

OBJECT: totalCharsIn { ports 2 }

Syntax: Counter

Definition: Total number of characters received on all serial ports.

OBJECT: totalCharsOut { ports 3 }

Syntax: Counter

Definition: Total number of characters transmitted on all serial ports.

OBJECT: totalErrsParity { ports 4 }

Syntax: Counter

Definition: Total number of parity errors detected on all serial ports.

OBJECT: totalErrsOverrun { ports 5 }

Syntax: Counter

Definition: Total number of overruns on all serial ports.

OBJECT: totalErrsFraming { ports 6 }

Syntax: Counter

Definition: Total number of framing errors on all serial ports.

OBJECT: portTable { ports 7 }

Syntax: Sequence of portEntry

Definition: Table of port entries; one entry for each port.

OBJECT: portEntry { portTable 1 }

Syntax:

```
::= Sequence {
    portIndex          Integer,
    iSpeed             Integer,
    oSpeed             Integer,
    ctrlLines          Integer,
    flowTypeIn         Integer,
    flowTypeOut        Integer,
    bitsPerChar        Integer,
    stopBits           Integer,
    parity             Integer,
    modemDCD           Integer,
    modemDTR           Integer,
    modemCTS           Integer,
    modemRTS           Integer,
    charsIn            Counter,
    charsOut           Counter,
    errsParity         Counter,
    errsOverrun        Counter,
    errsFraming        Counter,
    inCC               Gauge,
    outCC              Gauge,
}
```

Definition: The hardware characteristics of a port.

OBJECT: portIndex { portEntry 1 }

Syntax: Integer

Definition: The port number for each instance of these variables.

OBJECT: iSpeed { portEntry 2 }

Syntax: Integer {
50(1), 75(2), 110(3), 134(4), 150(4), 200(6)
300(7), 600(8), 1200(9), 1800(10),
2000(11), 2400(12), 3600(13), 4800(14),
7200(15), 9600(16), 19200(17), 38400(18),
idle(254), autobaud(255)
}

Definition: Input line speed for the port.

OBJECT: oSpeed { portEntry 3 }

Syntax: Integer {
50(1), 75(2), 110(3), 134(4),150(4), 200(6)
300(7), 600(8), 1200(9), 1800(10),
2000(11), 2400(12), 3600(13), 4800(14),
7200(15), 9600(16), 19200(17), 38400(18),
idle(254), autobaud(255)
}

Definition: Output line speed for the port.

OBJECT: ctrlLines { portEntry 4 }

Syntax: Integer {
none(1), dcddtr(2), ctsrts(3), both(4)
}

Definition: Type of hardware control used on the port: modem control (dcddtr), flow control (ctsrts), or both (available only on an Annex IIe or Annex 3).

OBJECT: flowTypeIn { portEntry 5 }

Syntax: Integer {
none(1), xonxoff(2), eia(3), bell(4)
}

Definition: Type of input flow control on the port.

OBJECT: flowTypeOut { portEntry 6 }

Syntax: Integer {
none(1), xonxoff(2), eia(3), bell(4)
}

Definition: Type of output flow control on the port.

OBJECT: bitsPerChar {portEntry 7 }

Syntax: Integer {
five(1), six(2), seven(3), eight(4)
}

Definition: Number of data bits in a character, not including the start, stop, or parity bits.

OBJECT: stopBits { portEntry 8 }

Syntax: Integer {
one(1), onefive(2), two(3)
}

Definition: Number of stop bits.

OBJECT: parity { portEntry 9 }

Syntax: Integer {
none(1), even(2), odd(3), mark(4), space(5)
}

Definition: Type of parity used on the port.

OBJECT: modemDCD { portEntry 10 }

Syntax: Integer {
unused(1), lo(2), hi(3)
}

Definition: The state of DCD on the port.

OBJECT: modemDTR { portEntry 11 }

Syntax: Integer {
unused(1), lo(2), hi(3)
}

Definition: The state of DTR on the port.

OBJECT: modemCTS { portEntry 12 }

Syntax: Integer {
unused(1), lo(2), hi(3)
}

Definition: The state of CTS on the port.

OBJECT: modemRTS { portEntry 13 }
Syntax: Integer {
unused(1), lo(2), hi(3)
}
Definition: The state of RTS on the port.

OBJECT: charsIn { portEntry 14 }
Syntax: Counter
Definition: Total number of characters received by the port.

OBJECT: charsOut { portEntry 15 }
Syntax: Counter
Definition: Total number of characters transmitted by the port.

OBJECT: errsParity { portEntry 16 }
Syntax: Counter
Definition: Total number of parity errors for the port.

OBJECT: errsOverrun { portEntry 17 }
Syntax: Counter
Definition: Total number of overruns for the port.

OBJECT: errsFraming { portEntry 18 }
Syntax: Counter
Definition: Total number of framing errors for the port.

OBJECT: inCC { portEntry 19 }
Syntax: Gauge
Definition: Current input buffer use in characters.

OBJECT: outCC { portEntry 20 }
Syntax: Gauge
Definition: Current output buffer use in characters.

Chapter 1

na Commands

General

The network administrator program, **na**, is a UNIX utility that provides the commands for managing the Annex Communications Server. These commands allow you to: set and display the operating characteristics of the Annex and its ports; reboot or reset the Annex and its ports; and broadcast administrative messages to the Annex ports. The Annex stores the parameters set using **na** in non-volatile memory. The **na** program can communicate with the Annex only when it is running its operational code. After a reboot or a reset, the Annex updates information modified using **na**.

All **na** commands are taken from its standard input. You can either run **na** interactively, or with its input coming from a file or pipeline. You can create a file containing **na** commands, called an **na** script file, to configure an Annex. The script file can also save the configuration information for a specific Annex and, when required, use **na** to restore the configuration. Script files are discussed in more detail later in this chapter.

Command Notation

Interactive **na** sessions allow you to enter **na** commands with or without arguments or parameters. If you enter the command without arguments or parameters, **na** prompts you for them. The conventions for interactive session are:

- You can abbreviate commands and parameter names to the minimum number of characters that uniquely distinguish the name from any other name that may appear in the same context.
- When entering a command with its arguments, a new line ends a command unless preceded immediately by the backslash character (\).
- When using a space as an argument, it must be enclosed in double quotes (""); otherwise the space is assumed to be a delimiter.
- The UNIX interrupt character (usually CTRL-C) returns you to the **command:** prompt.

Additionally, **na** permits comments when the # character is present at the beginning of a comment line. Any characters between the # and the next new line are ignored. The **na** commands use the following variables as arguments:

annex_identifier	A symbolic name or an Internet address assigned to an Annex. For example: `backhall` or `132.245.6.38` or `0xC0.0x9.0xC8.0x64`
annex_list	A list of one or more *annex_identifiers* separated by commas. For example: `backhall, 132.245.6.42, frontlobby`
port_identifier	A serial port number, a list of port numbers separated by commas, or a range of port numbers separated by hyphens. When followed by an @ and a *annex_identifier*, specifies a port for a specific Annex. For example: `5@132.245.6.42` or `1,3,5@thirdfloor` or `6-9@backhall`
port_set	A list of one or more *port_identifiers* separated by semicolons. A *port_set* can include ports on different Annexes. For example: `5@132.245.6.42; 1-8@thirdfloor`

annex_parameters	A list of one or more Annex parameters and values separated by white space (space, tab newline). For example:
	`pref_load_addr 132.245.6.66`
	`pref_dump_addr 132.245.6.66`

port_parameters	A list of one or more serial port parameters, with or without values. For example: `input_flow_control eia`
printer_parameters	A list of one or more parallel printer port parameters, with or without values, separated by white space. For example: `map_to_upper N`

This chapter illustrates command names, keywords, and parameter names in their long forms. Examples of **na** commands sometimes appear without the interactive **command:** prompt, and with embedded comments that describe the functions being performed. This format resembles the appearance of **na** scripts; the portion of the script entered at the terminal in response to the **command:** prompt is highlighted.

Commands

This section describes the following **na** commands:

annex	Defines a default *annex_list* used with subsequent commands.
boot	Boots the Annex.
broadcast	Sends a broadcast message to one or more ports.
copy	Copies the Annex, port, and printer configuration parameters to other Annex, ports, and printers.
dumpboot	Boots the Annex and produces a dump.
echo	Writes the remainder of the line to the standard output.
help or ?	Displays help for commands and parameters.
password	Defines a default administrative password used to communicate with an Annex.
port	Defines a default *port_set* used with subsequent commands.
quit	Terminates **na**.
read	Reads and executes a script file.
reset	Resets the Annex's subsystems or a *port_set*.
set	Defines or modifies the value of a parameter.
show	Displays the current value of a parameter.
write	Writes the current configuration to a script file.

After installing the **na** program on a UNIX host, type **na** at a terminal connected to this host. There are no arguments or command line options available. After entering **na**, the prompt **command:** appears. For example:

```
% na
Annex network administrator R6.0 March 3, 1991
command:
```

Seven of the **na** commands use standard UNIX superuser protection — only a superuser at the host can execute these commands: **boot, broadcast, copy, dumpboot, read, reset,** and **set.**

annex

The **annex** command establishes a default *annex_list* that is used in subsequent commands. Before issuing an **na** command, specify the Annex to which the executed command refers. The Annexes that you specify using the **annex** command become the default *annex_list*. You can group several Annexes into a single list, and then issue one command for the entire group of Annexes. The command syntax is:

annex *annex_list*

The following example creates an *annex_list* containing one Annex with the Internet address 132.245.6.40:

```
command: annex 132.245.6.40
```

The next example creates an *annex_list* containing two Annexes: one specified by its Internet address, and the other specified by its name:

```
command: annex 132.245.6.40, frontlobby
```

This example shows how **na** prompts for missing arguments:

```
command: annex
enter default annex list: 132.245.6.40, frontlobby
```

The **annex** command displays a message identifying the Annex, its Internet address, the number of serial lines, and the software version for each Annex in the *annex_list*. For example:

```
command: annex 132.245.6.1
132.245.6.1: Annex 3 R6.0, 64 ports
```

The **annex** command causes Annexes with security enabled and an administrative password set to prompt for that password. Alternately, you can use the **password** command to define a default password. If this default matches the Annex password, no password prompt appears and normal processing continues. The following example shows how to enter an administrative password:

```
command: annex frontlobby
Password for 132.245.6.40 <frontlobby>:
frontlobby: Annex-3 R6.0, 64 ports
```

The password is not echoed when entered using the **annex** command. If you enter an incorrect password, **na** prompts for the correct one. If the password is incorrect a second time, **na** drops the Annex from the *annex_list*. If an Annex in the list does not respond, **na** ignores that Annex and prints a status message:

```
132.245.6.1: Not responding
Warning:132.245.6.1 has been dropped from the list
```

The **na** program drops an Annex from the *annex_list* if:

- Its name could not be translated to an Internet address.
- The Annex does not respond because it is down.
- The wrong Internet address was entered using the **annex** command.

boot

The **boot** command reboots all Annexes in the *annex_list* and, optionally, produces a dump of the Annex's operational code. You can set a time at which the boot is to take place. The **boot** command can send a warning message to users attached to the Annex.

Note: When the Annex reboots, it terminates all active connections.

The command syntax is:

boot [-adhq] <*time*> <*annex_list*> <*filename*> <*warning*>

Supported arguments are:

-a Aborts any delayed boots that are pending.

-d Performs a dump before rebooting.

-h Performs a diagnostic boot using the ROM Monitor **boot** command if the Annex is in DIAG mode; see the *Annex Hardware Installation Guide*.

-q Performs a boot without sending a warning message.

You have two options for entering the boot time. The syntax is:

[+] [*HH*:] [*MM*]

HH:MM Indicates an exact clock time for the boot. For example: `15:15` for 3:15 p.m.

+HH:MM Indicates the number of hours and/or minutes before the boot takes place. For example: `+2:15` indicates a boot will take place in two hours and fifteen minutes.

If you do not include an *annex_list*, the command prompts for it. Pressing the **Return** key accepts the default *annex_list*.

The *filename* argument identifies the name of the Annex's operational code file. If you do not enter a *filename*, the command prompts you for one. If you press the **Return** key at that prompt, the Annex boots the default *filename*.

The *warning* argument allows you to enter an additional 256-character message. Warning messages are sent out to users periodically. If you do not specify a time delay or message, the **boot** command generates an automatic warning message.

The following example of the **boot** command requests a boot in one hour and fifteen minutes:

```
command: boot +1:15
    annex list (return for default): thirdfloor,132.24.5.6.40
    filename (return for default): RET
    warning: Shutting down for PM
```

The Annex can request its boot file from a defined preferred load host. If that host is not defined, or does not respond, the Annex broadcasts its request and boots from the first load host to respond.

broadcast

The **broadcast** command sends a message to specified ports at the identified Annexes. The command syntax is:

broadcast =[*port_set* | *keyword* [*@annex_identifier*]] *message*

The *port_set* argument indicates the port(s) to which the message is to be broadcast. The available keywords are:

all	Broadcasts to all serial ports and all virtual connections.
serial	Broadcasts to all serial ports.
virtual	Broadcasts to all virtual CLI connections (you cannot broadcast to a single virtual CLI connection).

If the *message* requires more than one line, using "\" at the end of each line inserts a new line.

copy

The **copy** command copies a given set of parameters to other sets of parameters. Parameter sets include:

annex	Copies all Annex parameters except the Internet address, the administrative password, the access control protocol key, LAT key, and the virtual CLI password from the specified Annex to the *annex_list.*
port	Copies all port parameters except the port password from the specified serial port to the *port_set.*
printer	Copies all printer parameters from the specified Annex to the *annex_list.*

Note: The **copy** command requires superuser privileges.

The command syntax is:

copy annex *annex_identifier annex_list*

copy port *port_number@annex_identifier port_set*

copy printer *@annex_identifier annex_list*

The following example illustrates copying port 1 parameters to the remaining parameters on the same Annex:

```
command: copy port 1@132.245.6.40 2-16@132.245.6.40
```

This example illustrates copying port 1 parameters from one Annex to port 5 on another Annex:

```
command: copy port 1@frontlobby 5@132.245.6.55
```

The next example illustrates copying printer parameters:

```
command: copy printer 132.245.6.34 frontlobby
```

dumpboot

The **dumpboot** performs a dump of every Annex specified in the *annex_list* immediately before it reboots the Annex. You can set the boot time, and the **dumpboot** command sends a warning message to users attached to the Annex.

Note: The **dumpboot** command requires superuser privileges. When the Annex **dumpboots**, it terminates all active connections.

The command syntax is:

dumpboot [-adhq] <*time*> <*annex_list*> <*filename*> <*warning*>

Supported arguments are:

-a Aborts any delayed dump boots that are pending.

-d Performs a dump before rebooting.

-h Performs a diagnostic boot using the ROM Monitor **boot** command if the Annex is in DIAG mode (see the *Annex Hardware Installation Guide*).

-q Performs a boot without sending a warning message.

You have two options for entering the boot time. The syntax is:

[+] [*HH*:] [*MM*]

HH:MM Indicates an exact clock time for the boot. For example: `15:15` for 3:15 p.m.

+*HH:MM* Indicates the number of hours and/or minutes before the boot takes place. For example: `+2:15` indicates a boot will take place in two hours and fifteen minutes.

If you do not include an *annex_list*, the command prompts for it. Pressing the **Return** key accepts the default *annex_list*.

The *filename* identifies the name of the Annex's operational code file. If you do not enter a *filename*, the command prompts you for one. If you press the **Return** key at the prompt, the Annex boots the default file name. The Annex requests the boot file from a preferred load host if it is defined and available; otherwise, it broadcasts a boot request.

The *warning* argument allows you to enter an additional 256-character message. Warning messages are sent to users periodically. If you do not specify a time delay or warning message, the **dumpboot** command generates an automatic warning message.

Following is an example of the **dumpboot** command:

```
command: dumpboot
    annex list (return for default): backhall
    filename (return for default): RET
    warning: Diagnostic testing
```

The Annex sends the dump to a defined preferred dump host. If that host is not defined or does not respond, the Annex broadcasts its dump request and dumps to the first host that responds.

echo

The **echo** command writes its argument to the standard output. This command is intended for use in script files. The **write** command automatically puts **echo** commands in the script file it writes. The *write* command section of this chapter includes an example of **echo** commands included in the script file created by the **write** command. The command syntax is:

echo *message*

help

The **help** command displays on-line help information about **na**. The command syntax is:

help [*command* | *parameter* | * | **syntax**]

? [*command* | *parameter* | * | **syntax**]

If you enter **help** alone, the Annex displays a list of **na** commands. If you enter **help syntax**, the Annex displays the command's syntax. If you enter **help ***, the Annex displays all available information on commands and parameters. If you enter the **help** command using another **na** parameter as its argument, the Annex displays the complete syntax for that command. For example:

```
command: help boot

boot (command):
boot [-adhq] <time> <annex_list> <filename> <warning>
```

If you enter the command using a parameter name as an argument, the Annex displays all the legal values for that parameter. For example:

```
command: help timezone_minuteswest

timezone_minuteswest (annex parameter):
Minutes west of GMT: an integer
```

Entering the full name of a command or parameter is not necessary. You can enter **help** followed by the first letter or first few letters of the command or parameter name. This displays all entries beginning with the string. For example:

```
command: help t

telnet_escape (serial port parameter):
escape character to use with the telnet command: a character

term_var (serial port parameter):
Terminal variable: a string, maximum sixteen characters

time_broadcast (annex parameter):
broadcast for time server to use if none found:
Y or y to enable; N or n to disable

timezone_minuteswest (annex parameter):
Minutes west of GMT: an integer

toggle_output (serial port parameter):
character used to toggle output: a character

type (printer parameter):
printer interface style: (dataproducts or centronics)


command: help te

telnet_escape (serial port parameter):
escape character to use with the telnet command: a character

term_var (serial port parameter):
Terminal variable: a string, maximum sixteen characters
```

password

The **password** command allows you to define a default password for that session of **na**. The command syntax is:

password [*password*]

The password is echoed when you enter it as an argument to the command. For example:

```
command: password edisraf
command:
```

If you enter the command without giving the password, you are prompted for it, but it is not echoed. For example:

```
command: password
Password: RET
```

When accessing an Annex with security enabled using the **annex** command, **na** will try to match the communication server's default password with the administrative password. If they match, access is authorized automatically; if they do not match, **na** prompts for the Annex-specific administrative password.

Enter a password for a given Annex only once during an **na** session, even if the Annex is dropped or the default *annex_list* is changed.

port

The **port** command establishes a default *port_set* used in subsequent commands until another port or a list of ports is specified. Grouping ports using a *port_set* allows you to issue one **na** command to examine or change the parameter values for multiple ports. The command syntax is:

port *port_set* | *keyword*

If you do not identify a specific Annex using the @ symbol and a name or Internet address when entering the *port_set*, all Annexes in the current *annex_list* are used. A *port_set* referring to the default *annex_list* is updated if a new Annex command is issued.

The keyword identifies groups of ports:

all	Sets default *port_set* to all serial ports and virtual CLI connections.
serial	Sets default *port_set* to all serial ports.

The following example defines port 1 on the Annex using the Internet address 132.245.6.34 as the default *port_set*:

```
command: port 1@132.245.6.34
```

The next example defines ports 1–5 on the same Annex as the default *port_set*:

```
command: port 1-5@132.245.6.34
```

The next example illustrates excluding port 4 from the range of ports 1–5 on the same Annex:

```
command: port 1-3, 5@132.245.6.34
```

This example defines all but port 6 on every Annex in the default *annex_ list*:

```
command: port 1-5, 7-16
```

This example illustrates defining ports on two different Annexes: ports 1–16 on the Annex at 132.245.6.34 and 1–8 on the Annex *backhall*:

```
command: port all@132.245.6.34; 1-8@backhall
```

quit

The **quit** command terminates the **na** program; **na** quits when it receives an end-of-file character (usually CTRL-D) or when it reaches the end of an input file. The command syntax is:

quit

read

The **read** command reads a script file that contains **na** commands. The **na** program executes these commands as if they were entered at a terminal in interactive mode. This command requires superuser privileges. The syntax is:

read *filename*

You can create script files using a text editor or the **write** command. Following is an example of a script file called **testscript** that modifies Annex parameters.

```
# standard parameters for Annexes on our network
set annex pref_load_addr 132.245.6.63
set annex pref_dump_addr 132.245.6.63
set annex load_broadcast Y
set annex name_server_1 dns
set annex pref_name1_addr 132.245.6.9
set annex host_table_size 30
set annex cli_prompt "%n%s%p%c"
set annex timezone_minuteswest 360
set annex daylight_savings usa
set annex enable_security Y
set annex vcli_security Y
set annex syslog_mask all
set annex syslog_host 132.245.6.9
```

Use this script as follows:

```
command: annex thirdfloor, frontlobby, backhall
command: read testscript
```

reset

The **reset** command changes some of the current attributes of all the Annexes in the default *annex_list* without rebooting them. Unless using the **reset** command, changes to **na** parameters for a specific port, virtual CLI connection, security, or name server become effective only after booting the Annex. The command syntax is:

reset [*port_set* | *keyword*]

The *port_set* resets the specified port(s). The keyword resets other Annex attributes. The available keywords are:

all	Resets all serial ports and virtual CLI connections.
annex all	Resets the message-of-the-day, the security, nameserver, and LAT subsystems, and customized user interface macros.
annex lat	Resets the LAT-specific Annex parameters so that any future LAT circuits (connections) will use the new values; existing circuits will continue to use the old values. This keyword will not terminate existing LAT circuits.
annex macros	Resets the customized user interface macros.
annex motd	Resets the message-of-the-day.
annex nameserver	Resets the name server parameters and flushes the Annex's host table.
annex security	Resets the security system.
printer	Resets the parallel printer port.
serial	Resets all serial ports.
virtual	Resets all virtual CLI connections.

Note: The **reset** command requires superuser privileges. A **reset** issued to **serial**, **virtual**, or a *port_set*, terminates any active connections.

An example of resetting the Annex's security subsystem and message-of-the-day is:

```
command: annex frontlobby
command: reset annex security
command: reset annex motd
```

The next example resets ports 1 through 8 on the Annex *frontlobby*:

```
reset 1-8@frontlobby
```

The following example resets all serial ports on the Annex *thirdfloor*:

```
reset all@thirdfloor
```

The last example resets the parallel printer port on the Annex *thirdfloor*:

```
reset printer@thirdfloor
```

set

The **set** command modifies Annex or printer port parameters:

set annex	Modifies Annex parameters.
set printer	Modifies parameters for the parallel printer port.
set port	Modifies parameters for the serial line ports.

The following commands require superuser privileges. The command syntax is:

set annex [=*annex_list*] *annex_parameters*

set port [=*port_set*] *port_parameters*

set printer [=*annex_list*] *printer_parameters*

The *annex_parameters*, *printer_parameters*, and *port_parameters* arguments require a name and a value separated by a space. A space is required between each parameter argument. You can enter more than one parameter arguments with each command. If you are entering multiple parameter arguments that require a new line, precede the new line with the "\" character. Changes made to parameters take effect after booting the Annex or resetting it or the port(s). An example of setting port parameters is:

```
command: set port speed 9600 data_bits 7 stop_bits 1\
         parity odd control_lines none type hardwired\
         mode cli inactivity_timer 120
```

show

The **show** command displays current Annex, printer, or port parameters:

show annex	Displays Annex parameters.
show port	Displays serial port parameters.
show printer	Displays parallel printer parameters.

The command syntax is:

show annex [=*annex_list*] [*keyword* | *annex_parameters*]

show port [=*port_set*] [*keyword* | *port_parameters*]

show printer [=*annex_list*] [*keyword* | *printer_parameters*]

The keyword specifies a subset of parameters for display. The supported keywords are:

all	Displays all Annex, port, and printer parameters. This is the default.
lat	Displays the LAT-specific Annex parameters.
interface	Displays all port parameters that define information about the interface, such as speed, parity, and number of data bits.
device	Displays all port parameters that define information about the attached device, such as type, mode, timers, and security.
editing	Displays all port parameters for use in CLI line editing. For example, the control character for erase character, erase word, erase line, etc.
slip	Displays all port parameters that define a serial port for SLIP use.

write

The **write** command creates a script file from the configuration data for a specific Annex. You can modify this script file using any text editor. The following *annex_parameters* are not included in the script created by the **write** command:

- Internet address
- Access control protocol key
- Administrative password
- Virtual CLI password
- LAT key
- Port password

The **write** command syntax is:

write *annex_identifier filename*

The **write** command writes **set annex, set printer,** and **set port** commands into the script file for each *annex_parameter*, *printer_parameter*, and *port parameter*. The **write** command also includes **echo** commands in the script file. When the script is executed using a **read** command, the arguments to the **echo** command are written to the standard output, indicating the progress of the read. Use the **write** command to copy Annex configuration data when installing new Annexes or making a back-up copy of an Annex's configuration data. For example:

```
command: write 132.245.6.101 fronthall.script
```

The following example uses the **write** and **read** commands to install a new Annex and to create a back-up copy of an Annex. The first line writes configuration data for the Annex *thirdfloor* to a file named **thirdfloor.prm**. The data from *thirdfloor* is copied to the new Annex specified in the *annex_list* defined using the **annex** command.

```
command: write thirdfloor thirdfloor.prm
command: annex 132.245.6.40
command: read thirdfloor.prm
```

Following is an excerpt from the script file **fronthall.script:**

```
# annex 132.245.6.101

echo setting annex parameters
set annex pref_load_addr 132.245.6.75
set annex pref_dump_addr 132.245.6.75
set annex load_broadcast Y
set annex image_name ""
set annex subnet_mask 255.255.255.0
set annex authoritative_agent Y
        :
        :
set annex routed Y
set annex rwho Y
set annex min_unique_hostnames Y
set annex ring_priority 0

echo setting printer parameters
set printer map_to_upper N
set printer printer_width 80
set printer hardware_tabs N
set printer type centronics

echo setting serial port parameters for port 1
set port=1 speed 19200
set port=1 data_bits 8
set port=1 stop_bits 1
set port=1 parity none
        :
        :
set port=64 slip_allow_dump Y
set port=64 slip_do_compression N
set port=64 slip_allow_compression Y
set port=64 slip_no_icmp Y
set port=64 slip_tos Y
```

Chapter 2

na Parameters

General

The **na** parameters define configuration data for a given Annex. This data includes information about the Annex, its parallel printer port, and its serial ports. The **set** command modifies these parameters. The **show** and **help** commands accept parameters as arguments. All parameters, except the Annex's Internet address, have a default. Use the **set** command to modify a parameter's default setting.

Parameter Conventions

This section describes the conventions for entering parameter values and returning those values to the supplied defaults.

Entering Parameter Values

The conventions for entering parameter values depend on the type of information the parameter defines.

- For parameters equiring an Internet address, specify the address in dot notation as a decimal number (from **0** to **255**), a hexadecimal number, or a combination of both. For example:

  192.9.200.100
  0xC0.0x9.0xC8.0x64
  192.9.200.0x64

- For parameters requiring a yes/no input, use either **Y** or **N**. These parameters are not case sensitive.
- For parameters that define passwords, the **na show** command displays only "**< set >**" or "**< unset >**"; it never displays the values entered for the parameters.
- Variable length strings allow a maximum of 16 characters, unless otherwise specified.

Resetting Parameters to Default Values

Resetting parameters to the supplied default values varies according to the parameter type. You can return any parameter value to its default, except the Annex's Internet address.

Resetting Annex Parameters

To reset *annex_parameters* to the default values, use the **set annex** command with one of the following commands:

1. **set annex** *annex_parameter* **0**
2. **set annex** *annex_parameter* **""**
3. **set annex** *annex_parameter* **default**

The first command, **set annex** *annex_parameter* **0**, resets parameters that require a numeric value, such as an Internet address, the **subnet_mask**, and the **network_turnaround**. To reset **pref_dump_addr** to its default, 0.0.0.0:

```
command: set annex pref_dump_addr 0
```

The second command, **set annex** *annex_parameter* **""**, resets all parameters that require a string. These parameters use either a null string ("") (e.g., the **image_name** parameter) or "**<unset>**" (e.g., the **password** parameter) for their default.

To reset **image_name** to its default:

```
command: set annex image_name ""
```

The third command, **set annex** *annex_parameter* **default**, resets the remaining parameters. To reset the **enable_security** value to its default, **N**:

```
command: set annex enable_security default
```

Resetting Printer and Port Parameters

To reset *port_parameters* and *printer_parameters* to the default values, use the **set port** or **set printer** command with one of the following commands:

1. **set port** *port_parameter* ^@
2. **set port** *port_parameter* ""
3. **set port** *port_parameter* **default**

or

1. **set printer** *printer_parameter* ^@
2. **set printer** *printer_parameter* ""
3. **set printer** *printer_parameter* **default**

The command **set port** *port_parameter* ^@ resets parameters that require a single-character value, such as **input_stop_char** or **erase_word**. Enter the default as a two-character sequence consisting of the circumflex character (^) followed by the at sign (@). The following example resets **erase_word** to its default, ^W:

```
command: set port erase_word ^@
```

The command **set port** *port_parameter* "" resets parameters that require a string. These parameters use a null string ("") (e.g., the **user_name** parameter) or **"<unset>"** (e.g., the **port_password** parameter) for the default. The following example resets the **user_name** to its default, a null string:

```
command: set port user_name ""
```

The command **set port** *port_parameter* **default** resets the remaining parameters. These parameters generally fall into two groups: those that require a choice from a known list or range, such as **speed** or **inactivity_timer**, and those that require a yes/no response, such as **allow_broadcast** or **short_break**. The keyword **default** resets these parameters to default values. The following example resets **speed** to its default, 9600:

```
command: set port speed default
```

To reset all of the Annex's parameters to the supplied defaults, use the ROM monitor **erase** command (see the *Annex Hardware Installation Guide*). This **erase** command erases all parameters, including the Annex's Internet address. After issuing this command, you ***must*** re-enter the Annex's Internet address and re-configure the Annex.

Parameter Descriptions

The **na** parameters are divided into four categories: 1) Annex parameters, 2) LAT-specific Annex parameters, 3) serial line port parameters, and 4) parallel printer port parameters.

Annex Parameters

This section describes the **na** parameters that define attributes of the Annex's operations. The **set annex** command modifies these parameters; the **show annex** command displays them.

acp_key

The **acp_key** parameter specifies a string for the Annex's encryption key for exchanging messages between the Annex and the security server. It is used with host-based security and has meaning only if the **enable_security** parameter is set to **Y**. The default is **"< unset >"**.

authoritative_agent

The **authoritative_agent** parameter enables the Annex to reply to an ICMP Address Mask Request. If a host broadcasts a request for the subnet mask, the Annex sends an ICMP Address Mask Reply. A **Y** enables this parameter, an **N** disables it. The default is **Y**.

broadcast_addr

The **broadcast_addr** parameter specifies the Internet address for Annex broadcasts. Set the broadcast address according to the network's broadcast scheme: *network.0* or *network.1*. The default is **0.0.0.0**.

Although the current Internet-recommended broadcast address is *network.1.*, some systems use the older scheme of *network.0*. A host configured with a *network.1* address will accept *network.0* broadcasts. Hosts configured with *network.0* addressing will not see *network.1* broadcasts.

cli_prompt

The **cli_prompt** parameter specifies the Annex prompt for all users that access the CLI. Specify a string consisting of characters and embedded formatting codes. The formatting codes consist of a percent character (%) followed by a single lower case character.

Each formatting code occupies one character in storage. You can also include a string with these codes for the prompt. The default prompt is **annex:**. Table D–1 lists and describes codes for the prompt string.

Table D–1. Formatting Codes for Annex Prompts

Code	Expansion
%a	The string **annex**.
%c	A colon followed by a space.
%d	The current date and time in the following format: Mon Mar 14 13:59:42 1991.
%i	The Annex's Internet address
%j	A new line character, skip to the beginning of the next line.
%l	The location defined for the port; if none, the string **port** *nn*,, where *nn* is the number of the serial line.
%n	The Annex's name, if known, or the Internet address.
%p	The port number, or the virtual CLI connection number in **v***n* form, where *n* is the virtual CLI connection number.
%r	The string **port**.
%s	A space.
%t	The current time in 24-hour format.
%u	The user name defined for the port; if none, a null string.

daylight_savings

The **daylight_savings** parameter defines the daylight savings time for your geographic location. The Annex uses this parameter to adjust the time display for daylight savings time. The options are **us, canadian, australian, british, west_european, east_european, mid_european,** and **none**; the default is **us.**

enable_security

The **enable_security** parameter enables/disables the security system. To enable any security features, host-based and/or local password protection, set this parameter to **Y**. The default is **N**.

host_table_size

The **host_table_size** parameter defines the number of entries in the host table. The range of allowable values is from **1** to **250**, **unlimited**, or **none**. The value **unlimited** sets no upper boundary on the size of the host table other than the amount of available memory. The value **none** indicates no host table. The default is **64**.

image_name

The **image_name** parameter specifies the file name that contains the Annex's operational code. The Annex assumes that this file is located in the directory **/usr/spool/erpcd/bfs**. The default is a **null string** ("").

inet_addr

The **inet_addr** parameter specifies the Annex's Internet address. This 32-bit address contains four 8-bit fields. Each field, separated by periods, is specified as either a decimal number ranging from **0** to **255**, or a hexadecimal number. The Internet address always displays in decimal notation. This parameter has no default.

ipencap_type

The **ipencap_type** parameter specifies whether the Annex LAN interface encapsulates IP packets in the Ethernet Version 2 format or the IEEE 802.2/802.3 Data Link Layer format. The values for this parameter are **ethernet** or **ieee802**. The default is **ethernet**.

Note: Original Annexes do not support this parameter.

lat_key

The **lat_key** parameter enables and disables the LAT-specific Annex parameters as well as the LAT protocol; it is a security mechanism that restricts access to LAT within the Annex. LAT commands, parameters, and functions are enabled only when the correct key has been set; after setting the key, the administrator *must* reboot the Annex. Each Annex requires its own unique key value to enable LAT (contact your supplier to obtain a LAT key).

Note: To manipulate LAT parameters after booting LAT for the first time, the network administrator must re-enter **na** or re-select the Annex.

load_broadcast

The **load_broadcast** parameter defines whether the Annex broadcasts for **gateways**, **rotaries**, message-of-the-day, or the **macros** file during a boot if any or all of the files were not available on the preferred load host. A **Y** enables this parameter, an **N** disables it. The default is **Y**.

load_dump_gateway

The **load_dump_gateway** parameter specifies the gateway's Internet address. This gateway is required if the preferred load or dump host is on a different network or subnet. The default is **0.0.0.0** (no gateway).

Note: Original Annexes do not support this parameter.

load_dump_sequence

The **load_dump_sequence** parameter specifies available network interfaces (Ethernet, token ring, or SLIP) and the order to be used for a down-line load or an up-line dump.

For a local area network, use **net**. For a SLIP line, use **sl***nn*, where *nn* is the number of the serial port. The default is **net**. You can enter up to four interfaces using this parameter, with each interface separated by a comma. For example:

```
set annex load_dump_sequence sl2,sl6,net
```

Note: Original Annexes do not support this parameter.

max_vcli

The **max_vcli** parameter determines the maximum number of virtual CLI connections the Annex can create at any one time. Allowable values are the string **"unlimited"** or a decimal number from **0** to **254**. A value of **0** prevents any virtual CLI connections. The default is **"unlimited"** and permits an unlimited number of virtual CLI connections.

min_unique_hostnames

The **min_unique_hostnames** parameter defines whether the *minimum uniqueness* feature is available. This feature allows you to enter the host name with a minimal string that uniquely identifies that host from any other host in the host table. A **Y** enables this parameter, an **N** disables it. The default is **Y**.

motd_file

The **motd_file** parameter defines the file name for the message-of-the-day file maintained on the load host. The default file name is **motd** and must reside in the directory **/usr/spool/erpcd/bfs**.

name_server_1

The **name_server_1** parameter defines the name service to use with the primary name server. When using this parameter, you must specify a host using the **pref_name1_addr**. The options are **bind, dns, ien_116,** and **none**. The options **bind** and **dns** are synonymous. The default is **none**.

name_server_2

The **name_server_2** parameter defines the name service to use with the secondary name server. The type specified with this parameter is queried if the type specified with **name_server_1** is not available. When using this parameter, you must specify a host using **pref_name2_addr**. The options are **bind, dns, ien_116,** and **none**. The options **bind** and **dns** are synonymous. The default is **none**.

nameserver_broadcast

The **nameserver_broadcast** parameter defines whether or not the Annex broadcasts a name server request if the preferred name servers do not respond. A **Y** enables this parameter, an **N** disables it. The default is **N**.

network_turnaround

The **network_turnaround** parameter specifies the amount of time the Annex waits for a response from a security server. It is used with host-based security and has meaning only if the **enable_security** parameter is set to **Y**. Specify this value in seconds from **1** to **10**. The default is **2** seconds.

password

The **password** parameter modifies the Annex's administration password. The default administrative password is the Annex's Internet address.

This password is used for access to the superuser CLI commands and for administrative access to an Annex. It overrides the CLI **lock** and virtual CLI passwords.

The **show annex** command does not display the password; it displays "**< set >**" or "**< unset >**". The default is "**< unset >**". After setting the password, it never displays again. If you forget this password, you can reset it only by erasing all of the Annex's non-volatile memory using the ROM monitor.

Note: Even if the **password** parameter displays as "**< unset >**", the default administrative password (the Annex's Internet address) is required to access the superuser CLI commands.

pref_dump_addr

The **pref_dump_addr** parameter specifies the Internet address for the preferred dump host. This is the host to which the Annex first tries to dump. The default is **0.0.0.0**.

pref_load_addr

The **pref_load_addr** parameter specifies the Internet address for the preferred load host. This is the host to which the Annex first requests a load of its operational code. The default is **0.0.0.0**.

pref_name1_addr

The **pref_name1_addr** parameter defines the Internet address of the host providing the name server specified in the **name_server_1** parameter. The default is **0.0.0.0**.

pref_name2_addr

The **pref_name2_addr** parameter specifies the Internet address of the host providing either the name server specified in the **name_server_2** parameter, or a back-up host if that parameter is set to **none**. The default is **0.0.0.0**.

pref_secure1_host

The **pref_secure1_host** parameter specifies the Internet address of the security server to which the Annex first sends security requests. It is used with host-based security and has meaning only if the **enable_security** parameter is set to **Y**. The default is **0.0.0.0**.

pref_secure2_host

The **pref_secure2_host** parameter specifies the Internet address of the host that is the back-up server if the host specified in **pref_secure1_host** is not available. Use this parameter with host-based security; it has meaning only if the **enable_security** parameter is set to **Y**. The default is **0.0.0.0**.

ring_priority

The **ring_priority** parameter, available on Annexes running IEEE 802.5 networks only, sets the priority for Annex-originated messages on a token ring LAN. The acceptable values range from **0** to **3**. The default is **0**.

routed

The **routed** parameter determines whether or not the RIP-listener of the routing daemon is enabled. A **Y** enables the parameter, an **N** disables it. The default is **Y**.

rwho

The **rwho** parameter determines whether or not the Annex listens for RWHO broadcasts when building its host table. A **Y** enables the parameter, an **N** disables it. The default is **Y**.

security_broadcast

The **security_broadcast** parameter determines whether or not the Annex broadcasts for security validation if the preferred security servers are not available. A **Y** enables the parameter, an **N** disables it. The default is **Y**.

server_capability

The **server_capability** parameter defines the Annex as a file server host. One Annex can provide operational code only for another Annex with the same hardware platform. The default is **none**. Allowable values are:

all	Copies of the operational code, and the **gateways**, **rotaries**, message-of-the-day, and **macros** files.
gateways	A copy of the **gateways** file.
image	A copy of its operational code.
macros	A copy of the **macros** file.
motd	A copy of the message-of-the-day file.
none	The Annex is not a file server.
rotaries	A copy of the **rotaries** file.

subnet_mask

The **subnet_mask** parameter specifies the Annex's Internet subnet mask. It is required only if the network is divided into subnets. Define the network address, and the subnet address, using the decimal value of 255 in the subnet mask. See *Book A: Overview, Chapter 2: Network Protocols*, for more details on subnets and subnet masks. The default for the subnet mask is based on the network part of the Annex's Internet address.

syslog_facility

The **syslog_facility** parameter defines the facility used in the Annex syslog messages. Specify the parameter as **log_local***n*, where *n* is a number from **0** through **7**. The default is **log_local7**. If the host to which messages are logged does not support 4.3BSD syslogging, **syslog_facility** is ignored and messages are logged by priority level (defined by the **syslog_mask** parameter).

syslog_host

The **syslog_host** parameter defines the Internet address of the host configured to log Annex messages. The default, **0.0.0.0**, causes the Annex to broadcast its log messages.

syslog_mask

The **syslog_mask** parameter determines the priority levels that are to be logged. The possible values are **all**, **none**, or a combination of levels separated by commas. The default, **none**, disables logging. See *Book B: Configuration Procedures, Chapter 1: Configuring the Annex*, for more details on using syslog for event logging. The levels in priority order are:

emergency	Hardware failures.
alert	All Annex reboots.
critical	Configuration and initialization problems, such as format errors in the **gateways** file, or lack of memory.
error	All line initialization errors, including CLI.
warning	Indications of minor problems.
notice	Time server queries and information about responses.
info	Start and end CLI sessions and Annex jobs created by the **rlogin, telnet, connect, ping**, and **tap** commands.
debug	Activate and exit all Annex processes.

system_location

The **sys_location** parameter supplies host location or identification information. This string allows a maximum of 32 characters. The default value is **null.**

tftp_dump_name

The **tftp_dump_name** parameter provides the name of the file to use for dumping the core image via TFTP if the Annex operational image and the ERPC protocol mechanism fail. The **tftp_dump_name** includes the entire path of the dump file including parent directories. This file must exist with read/write permissions to permit the Annex to open via TFTP (depends on the host's TFTP implementation).

tftp_load_dir

The **tftp_load_dir** is the string prepended to the image name, **gateways, rotaries, macros,** and message-of-the-day file names for TFTP transfers. The appropriate value of this string is determined by the system serving the TFTP requests. This string is ***not*** prepended to the **tftp_dump_name.**

time_broadcast

The **time_broadcast** parameter defines whether the Annex broadcasts for the time if the preferred load host is not available or does not provide a time server. A **Y** enables this parameter, an **N** disables it. The default is **N.**

timezone_minuteswest

The **timezone_minuteswest** parameter defines the time zone in which the Annex resides. Enter a positive number of minutes for time zones west of GMT, and a negative number for time zones east of GMT. For example, since Eastern Standard Time is west of GMT, it uses a value of 300 minutes; Paris is east of GMT and uses a value of –60. The default is **300.**

vcli_password

The **vcli_password** parameter defines a password required for virtual CLI connections to the Annex. This parameter is useful for local password protection and as a back-up to host-based security. For local password protection, set the **enable_security** parameter to **Y**, set the **vcli_security** parameter to **N**, and define a password for this parameter. As a backup for host-based security, setting this parameter causes the Annex to request a password on a virtual CLI connection whenever the security server does not respond. The default is "**< unset >**".

Note: Original Annexes do not support this parameter.

vcli_security

The **vcli_security** parameter enables/disables user validation on virtual CLI connections to an Annex. If this parameter is set to **Y**, the Annex executes the same user validation, including user name and password, that is used with CLI security on the serial ports. The default is **N**.

The **vcli_security** parameter is used with host-based security and has meaning only if the **enable_security** parameter is set to **Y**.

Note: Setting **vcli_security** to **Y** has the same effect as enabling **cli_security** and **connect_security** on a serial port.

Original Annexes do not support this parameter.

LAT-specific Annex Parameters

The LAT-specific Annex parameters are visible only when the **lat_key** parameter contains the correct key value for the Annex. These parameters provide some LAT protocol control, limits, and identification. The **reset annex lat** command resets these parameters; the **show annex lat** command displays them.

The LAT parameter values are not implemented until you either reboot the Annex or issue the **reset annex lat** command at the **na** prompt. If you reset the LAT parameters, the new values take affect for new LAT circuits only. The existing LAT circuits continue to use the old values until they are closed.

circuit_timer

The **circuit_timer** parameter is the time interval in tens of milliseconds between the transmission of LAT packets. Allowable values range from **1** to **100**. The default value is **8** (80 milliseconds).

facility_num

The **facility_num** also is referred to as the host number. Allowable values range from **0** to **32767**. The default value is **0**.

group_value

The **group_value** parameter is a security mechanism that restricts access to LAT services for all users on the Annex. There are 256 group codes, each of which is enabled or disabled. The default is **all disabled**.

keep_alive_timer

The **keep_alive_timer** parameter is the time interval, in seconds, between the transmission of identification packets during times of network inactivity. The packets serve only as notices to remote nodes that the host's services are available. Allowable values range from **10** to **255**. The default is **20 seconds**.

retrans_limit

The **retrans_limit** parameter is the number of times to retransmit a packet before notifying the user of a network failure. Allowable values range from **4** to **120**. The default value is **8**.

server_name

The **server_name** is a string of characters used to name the Annex in the LAT protocol. This name should be unique within your local network. Allowable string values are less than or equal to 16 characters. The default value is the physical ethernet address, represented as a hexadecimal value, appended to the string **LAT_**, For example, **LAT_080002BF0020**.

service_limit

The **service_limit** parameter is the upper bound on the number of services that the Annex can maintain in its local service table. Once the services table is full, new service entries may not be retained. The Annex tries to remove entries when the services table is full, based on an aging mechanism: the service that has been idle longest is removed from the service table; if all services are busy, and the services table is full, the new service is discarded. Allowable values range from **16** to **2048**. The default value is **256**.

Serial Line Port Parameters

The **set port** command modifies serial line port parameters; the **show port** command displays them. Additionally, the **show port** command provides four keywords (**interface**, **device**, **editing**, and **slip**) that display a specific subset of port parameters.

allow_broadcast

The **allow_broadcast** parameter allows the port to receive administrative broadcast messages generated by the **na broadcast** command and the **na** and CLI boot commands. A **Y** enables this parameter, an **N** disables it. The default is **Y**.

attn_char

The **attn_char** parameter defines a control character sequence as an attention character. Pressing the attention key returns you to the CLI prompt. The default is no attention character, displayed as ^@; the default for virtual CLI connections is **CTRL-A**.

The attention key is used with virtual CLI connections to the Annex, with modems that do not send a break properly, and with function keys to program interrupts. Users can define their own attention character using the CLI **stty attn** command.

bidirectional_modem

The **bidirectional_modem** parameter configures the port for adaptive use with a bidirectional modem. When enabled, the Annex allows connections to the port without waiting for the Carrier Detect signal. A **Y** enables this parameter, an **N** disables it. The default is **N**.

broadcast_direction

The **broadcast_direction** parameter defines the direction, either **network** or **port**, that an administrative broadcast message is sent on a port. This parameter is valid only when the port has been defined as **slave** using the **mode** parameter.

- If you specify **network**, the Annex sends administrative broadcast messages out the network side of the connection to the initiator. For example, if a person is connected to an attached modem, all administrative broadcast messages are sent back to the person who made the port connection.
- If you specify **port**, the Annex sends broadcast messages out the port side of the connection. The default is **port**.

char_erase

When the **char_erase** parameter is enabled, the Annex echoes both the character erase and the word erase characters for a video terminal; i.e., the previous character (or word) looks as if it has been erased. A **Y** enables this parameter, an **N** disables it. The default is **Y**.

When **char_erase** is disabled, the Annex echoes the erase characters for a hard-copy terminal. It echoes the first erase character as a "\" followed by the deleted character. Each additional use of the erase character deletes and displays another character. The first character typed (other than the erase character) echoes a "/ " and the character. For example, typing **asdf< Delete Delete >g** echoes as **asdf\fd/g**.

cli_inactivity

The **cli_inactivity** parameter specifies the amount of time in minutes that the Annex waits before hanging up the port after it becomes idle. Unlike the **inactivity_timer**, this timer does not disconnect a user's open session. The **cli_inactivity** timer effects only ports with no CLI active jobs.

Allowable values range from **0** (or **off**) to **254** minutes, or **immediate**. Specifying **immediate** causes the Annex to hang up the port immediately after exiting from the last job. The default is **off** (timer disabled).

cli_security

The **cli_security** parameter enables security for an Annex. This parameter is mandatory if you intend to use any Annex security mechanisms except the administrative password for accessing administrative tools. A **Y** enables this parameter, an **N** disables it. The default is **N**.

connect_security

The **connect_security** parameter enables the host-based security policy for access from the CLI to the network. If enabled, the user must receive authorization to connect to a host on the network. A **Y** enables this parameter, an **N** disables it. The default is **N**.

The supplied security policy scans the file **/etc/acp_restrict** to authorize a connection to a host from the Annex. If authorization is not granted, the connection is not made.

control_lines

The **control_lines** parameter defines the type of hardware control lines used on this port. The default is **none**. Allowable values are:

both	Allows both flow control and modem control on the port; available only on the Annex IIe or Annex 3.
flow_control	Configures the control lines for flow control (RTS/CTS), but does not activate hardware flow control: set the serial line parameters **output_flow_control** and/or **input_flow_control** to **eia**.
modem_control	Configures the control line for modem control using DTR/DCD/DSR hand shaking.
none	Does not implement hardware control lines.

data_bits

The **data_bits** parameter defines the number of data bits in a character. This value does not include the start, stop, or parity bits. Allowable values are **5–8**. The default is **8**.

dedicated_address

The **dedicated_address** parameter, along with the **dedicated_port** parameter, defines one host and one TCP port to which this port can connect. To use this parameter, set the **mode** parameter to **dedicated**. The **dedicated_address** is the host's Internet address. The default is **0.0.0.0**.

dedicated_port

The **dedicated_port** parameter selects the application or TCP port to which this port can connect. Allowable values are **telnet, rlogin,** or a TCP port number. The default is **telnet**.

dptg_settings

The **dptg_settings** parameter is reserved for Xylogics' internal use. Do not change this parameter's setting.

echo

When the **echo** parameter is enabled, the Annex echoes all characters as they are typed. A **Y** enables this parameter, an **N** disables it. The default is **N**.

erase_char

The **erase_char** parameter defines the character erase character. The allowable value is a control character sequence. The default is the **Delete** key (displayed as **^?**). This parameter has effect only at the CLI level.

erase_line

The **erase_line** parameter defines the line erase character. The allowable value is a control character sequence. The default is **CTRL-U**. This parameter has effect only at the CLI level.

erase_word

The **erase_word** parameter defines the word erase character. The allowable value is a control character sequence. The default is **CTRL-W**. This parameter has effect only at the CLI level.

forwarding_count

The **forwarding_count** parameter controls the behavior of the Annex port for received characters. When set, the port will not forward received characters until it receives the specified number of characters. Allowable values range from **0** to **255**. The default is **0**.

The **forwarding_count** parameter should be used only in conjunction with the **forwarding_timer** parameter since typed data will not be echoed until the **forwarding_count** is hit. The **forwarding_timer** overrides the **forwarding_count** if the count is not hit before the timer expires.

forwarding_timer

The **forwarding_timer** parameter defines an inter-character timer that sets the amount of time, in tens of milliseconds, that will elapse before forwarding received data. If new data is received before the timer expires, the Annex resets the timer and defers sending the data. The allowable values range from **0** to **255**, or **off** (which disables the timer). The default for the Annex 3 is **5**; the default for earlier Annexes is **0**.

hardware_tabs

The **hardware_tabs** parameter allows the terminal to expand ASCII tab characters if the terminal does not support hardware tabs. A **Y** enables this parameter, an **N** disables it. The default is **Y**. This parameter has effect only at the CLI level.

imask_7bits

When the **imask_7bits** parameter is enabled, the Annex port ignores the eighth bit of received characters. This parameter has no effect on transmitted characters. A **Y** enables this parameter, an **N** disables it. The default is **N.**

inactivity_timer

The **inactivity_timer** specifies the amount of time in minutes a port can remain inactive. If the timer expires, all sessions are terminated and the port is reset. You can define *activity* as either input to the port or output from the port using the **input_is_activity** and **output_is_activity** parameters. Allowable values range from **0** to **255**, or **off** (which disables the timer). The default is **off.**

input_buffer_size

The **input_buffer_size** parameter allocates available per port memory, in 512-byte blocks, preventing overflow from devices that send large amounts of data to the Annex.

Each 512-byte block holds 256 characters; status information occupies the remaining space. Allowable values range from **1** to **255**. Since the minimum value is **4**, values from **1** to **4** have the same effect as entering **4**. The default is **4** blocks.

input_flow_control

The **input_flow_control** parameter specifies the method the Annex uses to stop input from the terminal if its input buffer is about to overflow. The default is **bell.** Allowable values are:

bell	The Annex rings the terminal bell when its input buffer is full.
eia	Selects hardware flow control and only works if the **control_lines** parameter is enabled for hardware flow control, and the device is wired appropriately.
none	Specifies no flow control; characters are lost if the buffers overflow.
start/stop	Specifies XON/XOFF flow control. The Annex sends XOFF when its buffers are nearly full and sends XON when the buffer level reaches a safe level.

The Annex does not accept more input until it processes some of the characters in the buffer, unless the input is one of the erase characters.

input_is_activity

The **input_is_activity** parameters defines activity as input. If enabled, the Annex resets the inactivity timer when input is received at the port. A **Y** enables this parameter, an **N** disables it. The default is **Y**.

input_start_char

The **input_start_char** parameter defines the control character sequence that restarts input if the **input_flow_control** parameter is set to **start/stop**. The default is **CTRL-Q**.

input_stop_char

The **input_stop_char** parameter defines the control character sequence that stops input if the **input_flow_control** parameter is set to **start/stop**. The default is **CTRL-S**.

ixany_flow_control

The **ixany_flow_control** parameter treats any input character as a start (XON) character, if output has been suspended by a stop (XOFF) character. A **Y** enables this parameter, an **N** disables it. The default is **N**.

leap_protocol_on

The **leap_protocol_on** parameter allows you to use the Local Editing Accelerator Protocol (LEAP). A **Y** enables this parameter, an **N** disables it. The default is **N**.

line_erase

When the **line_erase** parameter is enabled, the Annex echoes line erase for a video terminal. It erases all characters on the line and moves the cursor back to the beginning of the line. A **Y** enables this parameter, an **N** disables it. The default is **Y**.

When **line_erase** is disabled, the Annex echoes the line erase character for hard-copy terminals. It echoes the erase character and a new line, so that the deleted line is still visible, but the print head is positioned at the beginning of the next line.

location

The **location** parameter defines a string that represents a location or a similar descriptive item. The CLI **who** command displays this value. The default is a **null string.**

long_break

When the **long_break** parameter is enabled, the Annex returns the user to the CLI prompt after receiving a break greater than two seconds. A **Y** enables this parameter, an **N** disables it. The default is **Y.**

map_to_lower

The **map_to_lower** parameter enables case conversion for characters sent from the terminal to the Annex. The Annex converts typed upper case characters to lower case. Typically, this parameter is enabled for older terminals without lower case characters. A **Y** enables this parameter, an **N** disables it. The default is **N.**

map_to_upper

The **map_to_upper** parameter enables case conversion for characters sent from the Annex to the terminal. The Annex converts lower case characters sent to the terminal to upper case. A **Y** enables this parameter, an **N** disables it. The default is **N.**

max_session_count

The **max_session_count** parameter specifies the number of active sessions (jobs) allowed. The allowable values range from **1** to **16.** The default is **3.**

mode

The **mode** parameter sets the mode for access to a serial line port. This parameter determines whether access is initiated by a device to the Annex or from the network through the Annex to the device. The default is **cli**. Allowable values are:

cli	Provides access from a device to the CLI. This mode is used for ports connected to terminals, and to incoming modems for access to the CLI. From the CLI, the user has access to the network and can use connect commands (**telnet**, **connect**, and **rlogin**) to access other hosts.
slave	Provides access to a port through the port server. This mode provides login lines for hosts with no Ethernet interface through a port server, and to access modems and serial line printers. This mode does not provide access to the CLI.
adaptive	Provides a port with both **slave** and **cli** capability. If a connection is initiated on the serial side, the port enters **cli** mode; if the connection is accessed via a port server, the port enters **slave** mode..
dedicated	Provides a port that can communicate only with one specific host defined using the **dedicated_address** parameter.
slip	Provides a port that can perform as a network interface using SLIP. IP packets are encapsulated using SLIP.
unused	Indicates that the port has no connection; the Annex ignores it.

need_dsr

When using a modem connected to a slave port, if the **need_dsr** parameter is enabled, the connection fails if no DSR signal is present; if **need_dsr** is disabled, the Annex accepts the connection, but waits for DSR and DCD before communicating with the modem; if the **bidirectional_modem** parameter is enabled, the Annex only waits for DSR. A **Y** enables this parameter, an **N** disables it. The default is **N**.

newline_terminal

The **newline_terminal** parameter controls the echoing of carriage returns. If **newline_terminal** is set to **Y**, a carriage return echoes as a carriage return; when set to **N**, a carriage return echoes as a carriage return followed by a line feed. The default is **N**.

output_flow_control

The **output_flow_control** parameter specifies the method that the terminal uses to stop output from the Annex. The default is **start/stop**. Allowable values are:

bell	Comparable to setting the parameter to **none**.
eia	Selects hardware flow control; **eia** works only if the **control_lines** parameter is enabled for hardware flow control and the terminal is appropriately wired.
none	Specifies no flow control; characters are lost if the buffers overflow.
start/stop	Specifies XON/XOFF flow control. Upon receiving XOFF, the Annex stops sending output to the terminal. Upon receiving XON, the Annex starts sending output to the terminal.

output_is_activity

The **output_is_activity** parameter defines activity as output. If enabled, the Annex resets the inactivity timer when output is sent to the port. A **Y** enables this parameter, an **N** disables it. The default is **N**.

output_start_char

The **output_start_char** parameter defines the control character sequence that restarts output. The default is **CTRL-Q**.

output_stop_char

The **output_stop_char** parameter defines the control character sequences that stops output if the **output_flow_control** parameter is set to **start/stop**. The default is **CTRL-S**.

parity

The **parity** parameter defines the type of parity that the device uses. Allowable values are **even, odd,** or **none**. The default is **none**.

port_multiplex

The **port_multiplex** parameter is reserved for Xylogics' internal use. Always set **port_multiplex** to N.

port_password

The **port_password** parameter defines a password for the port for local password protection. This parameter can also be used as a back-up for host-based security if the security servers do not respond.

If security is enabled for the Annex and this parameter is set, you must enter this password to access the port whether the **mode** parameter for the port is set to **cli, adaptive,** or **slave.**

The **show port** command does not display the password; it displays "**<set>**" or "**<unset>**". The default is "**<unset>**".

Note: Original Annexes do not support this parameter.

port_server_security

The **port_server_security** parameter enables a host-based security policy for access to the port through the port server. If enabled, only authorized users can access the port. A **Y** enables this parameter, an **N** disables it. The default is **N.**

Note: If you implement the supplied security policy, the user must enter both the user name and the password for CLI security.

prompt

The **prompt** parameter defines a port-specific CLI prompt. The value for this parameter is a prompt string consisting of characters and embedded formatting code. The prompt string is expanded when the prompt is displayed.

Table D–1 in *Annex Parameters, cli_prompt*, lists and describes these codes. The default is the prompt defined using the **cli_prompt** parameter.

Note: Original Annexes do not support this parameter.

redisplay_line

The **redisplay_line** parameter defines the reprint line character. The allowable value is a control character sequence. The default is **CTRL-R**.

reset_idle_time_on

The **reset_idle_time_on** parameter defines whether **input** or **output** resets the idle timer. The idle time is the time lapse between activity and inactivity at the terminal.The default is **input**.

short_break

When the **short_break** parameter is enabled, the Annex returns the user to the CLI prompt after receiving a break shorter than two seconds. A **Y** enables this parameter, an **N** disables it. The default is **Y**.

slip_allow_compression

The **slip_allow_compression** parameter allows the Annex to use TCP header compression if the other end of the SLIP link also initiates compression. A **Y** enables this parameter, an **N** disables it. The default is **N**.

Note: Original Annexes do not support this parameter.

slip_do_compression

The **slip_do_compression** parameter starts TCP/IP header compression on this SLIP link. If this parameter is enabled, the Annex always negotiates for TCP/IP compression on each connection. A **Y** enables this parameter, an **N** disables it. The default is **N**.

Note: Original Annexes do not support this parameter.

slip_allow_dump

The **slip_allow_dump** parameter enables dumping an Annex across a SLIP link. A **Y** enables this parameter, an **N** disables it. The default is **Y**.

Note: Original Annexes do not support this parameter.

slip_load_dump_host

The **slip_load_dump_host** parameter defines the host from which the Annex receives a load or to which the Annex dumps over the SLIP interface. If the **load_dump_sequence** Annex parameter is set to **sl*nn***, you must define an Internet address using this parameter. This parameter is used in place of the **pref_load_host** parameter. The default is **0.0.0.0**.

Note: Original Annexes do not support this parameter.

slip_local_address

The **slip_local_address** parameter defines the Internet address for this port (the Annex's side of the point-to-point SLIP link). This Internet address is used only for the SLIP interface. The default is **0.0.0.0**.

Note: Original Annexes do not support this parameter.

slip_no_icmp

The **slip_no_icmp** parameter discards any ICMP packets directed to the SLIP link. This parameter allows the Annex to reduce unnecessary traffic and messages over the SLIP link. A **Y** enables this parameter, an **N** disables it. The default is **N**.

Note: Original Annexes do not support this parameter.

slip_metric

The **slip_metric** parameter defines the hop count to the remote end of the SLIP interface. Modify this parameter only if you want the Annex to use a route other than the SLIP interface to the remote end. The default is **0**.

Note: Original Annexes do not support this parameter.

slip_remote_address

The **slip_remote_address** parameter defines the Internet address for the host at the other end of the serial line. The default is **0.0.0.0**.

Note: Original Annexes do not support this parameter.

slip_subnet_mask

The **slip_subnet_mask** parameter defines the subnet mask for the SLIP interface. The default is **0.0.0.0**.

Note: Original Annexes do not support this parameter.

slip_tos

When the **slip_tos** parameter is enabled, the Annex sends interactive traffic (**telnet, rlogin,** and ftp control sessions) before any other traffic. This parameter provides a type-of-service queuing on the link. A **Y** enables this parameter, an **N** disables it. The default is **N**.

Note: Original Annexes do not support this parameter.

speed

The **speed** parameter defines the speed of the serial line between the device and the Annex. The value defined using this parameter must match the line's baud rate. Allowable values are **autobaud, 50, 75, 110, 134.5, 150, 200, 300, 600, 1200, 1800, 2000, 2400, 3600, 4800, 7200, 9600,19200,** or **38400.** If the **speed** is set to **autobaud,** the Annex sets the speed by analyzing the data pattern generated by the **Return** key. The default is **9600.**

stop_bits

The **stop_bits** parameter specifies the number of bit time intervals between successive characters. Allowable values are **1, 1.5,** or **2.** The default is **1.**

telnet_crlf

When the **telnet_crlf** parameter is enabled, a newline translates to <CR><LF>; when disabled, a newline translates to <CR><NULL>; the default is **disabled.**

telnet_escape

The **telnet_escape** parameter defines the character that returns the user to the **telnet:** prompt when the user runs the CLI **telnet** command. The default is **CTRL-]**.

term_var

The **term_var** parameter identifies the type of terminal using the CLI connection. The value should be a valid terminal type for the host. The Annex passes the terminal type to the host using the **telnet** command. The default is a **null string** ("").

toggle_output

The **toggle_output** parameter defines the flush character. Pressing this character flushes the output buffer. The default is **CTRL-O.**

type

The **type** parameter defines the type of device connected to the port. The default is **hardwired**. Allowable values are:

dial_in	A modem or a device that behaves like a modem.
hardwired	A directly connected serial device.

user_name

The **user_name** parameter defines a string that represents the user of the port. The CLI **who** command displays this value; the CLI **rlogin** command passes this value to a host. The default is a **null string** (**""**).

Parallel Printer Port Parameters

The parallel printer parameters define the characteristics for the parallel printer port. Do not use these parameters to define a serial line printer. The **set printer** command modifies these parameters; the **show printer** command displays them.

hardware_tabs

The **hardware_tabs** parameter enables the printer to use hardware tab stops. If this parameter is disabled, the Annex converts a tab character to the number of spaces necessary to reach the next tab stop. **Y** enables this parameter, an **N** disables it. The default is **N**.

map_to_upper

The **map_to_upper** parameter enables case conversion. This parameter is enabled for printers that do not support lower case characters. A **Y** enables this parameter, an **N** disables it. The default is **N**.

printer_width

The **printer_width** parameter sets the maximum number of characters per line. The default is **80**.

type

The **type** parameter defines whether the printer is using a Centronics or a Dataproducts interface. Allowable values are **centronics** or **dataproducts**. The default is **centronics**.

Note: This parameter is valid only with the Annex IIe and the Annex 3.

Chapter 3

CLI Commands

General

The Command Line Interpreter (CLI) is the command interface for the Annex. At the CLI, you enter commands that connect to hosts, manage jobs (or sessions), display and modify port parameters, and display information for the Annex and the network.

The CLI provides two groups of commands: user and superuser. You administer the Annex using the superuser commands.

Command Syntax

You can shorten any CLI command or host name to the minimum number of letters that make the name unique. This is referred to as *minimum uniqueness*. If you do not want the Annex to interpret a host name using minimum uniqueness, enclose the name in double quotes (""). For example, entering **hosts "new"** prevents ambiguities between hosts **newark** and **new**. You can enter commands and host names in all lower case, all upper case, or a combination of both. The Annex performs any necessary case conversion.

User CLI Commands

The *Annex Communications Server User's Guide* describes the user CLI commands in greater detail. The user CLI commands are:

?	Displays help information for user CLI commands.
bg	Puts a job in the background.
connect	Uses LAT to connect to an advertised LAT service.
fg	Returns to an established job.

hangup	Disconnects all jobs and resets user CLI connections.
help	Displays help information for user CLI commands.
hosts	Displays the current host table.
jobs	Displays a list of current jobs.
kill	Terminates a job.
lock	Locks a port.
netstat	Displays network statistics.
rlogin	Connects to a host using the **rlogin** protocol.
services	Displays the LAT services that have been advertised by LAT nodes.
slip	Converts a CLI port to a SLIP interface port.
stats	Displays Annex statistics.
stty	Displays and modifies CLI port parameters.
telnet	Connects to a host using the **telnet** protocol.
who	Displays Annex users.

bg

The **bg** (background) command puts a job in the background. The Annex forwards output generated by the host for the background job to the terminal unless you are at the CLI. The command syntax is:

bg [*argument*]

Valid **bg** arguments are:

none, **%**, **%%**, **%+**	Puts most recent job (+) to background
%-	Puts previous job (-) to background
n, *%n*	Puts job *n* to background
%hostname	Puts job at *hostname* to background

The **bg** command displays the job number and the CLI command that created it. The **&** indicates that the job is running in the background.

connect

The **connect** command uses the LAT protocol to connect to an advertised LAT service. This command is available only if LAT is enabled and the **lat_key** parameter is set correctly. The command syntax is:

connect *service* [*hostname* [*port*]]

If the service to which you are connecting requires a password, the CLI prompts you for one. Entering the **connect** command with only the desired service name causes the command to connect to the service of that name with the greatest rating. Entering the command with both the service name and the hostname overrides this distribution mechanism; the command tries to establish a connection to that service on the specified host. Entering the command with the service name, hostname, and port name causes the command to connect to the service name on the specified port on the specified host.

fg

The **fg** (foreground) command resumes a job that has been suspended or placed in the background. If the Annex saved any output from the host while the job was interrupted, the output appears on the terminal immediately after reconnecting. Otherwise, nothing appears until you provide keyboard input. The command syntax is:

fg [*argument*]

Valid **fg** arguments are:

none, **%**, **%%**, **%+**	Puts most recent job (+) to foreground
%-	Puts previous job (-) to foreground
n, *%n*	Puts job *n* to foreground
%hostname	Puts job at *hostname* to foreground

The **fg** command displays the job number and the CLI command that created it. Entering **%** is the same as entering **fg %**, and returns you to the most recent job.

hangup

The **hangup** command terminates all of your jobs at the Annex and resets the CLI for the port. The **hangup** command also restores the default terminal characteristics defined for the port. The command syntax is:

hangup

If you have any open jobs on the CLI, the **hangup** command lists them and prompts for permission to terminate each job by requiring a **yes** or **no** response before completing the command.

help

The **help** command provides on-line help. The command syntax is:

help [*command*]

Entering the **help** command without arguments displays a summary list of all available CLI commands and macros defined for the Annex. Entering the command using another CLI command as its argument (e.g., **help hosts**) displays a short description of the command and its syntax.

hosts

The **hosts** command displays the names and addresses of hosts and other Annexes listed in the Annex's host table. The **hosts** command also displays any status information that a host broadcasts. The command syntax is:

hosts [**-q**] [*host*]

Entering the **hosts** command without arguments displays a list of all known hosts. Entering a host name using the command (e.g., **hosts alpha**) displays information only about that host. You can also enter the minimum number of letters to identify a host, or group of hosts, if you have enabled the **min_unique_hostnames** parameter. For example, entering **hosts a** displays information for all host names beginning with the letter a. Entering the **-q** argument displays only the names of known hosts.

jobs

The **jobs** command displays information for all current jobs (or sessions). The command syntax is:

jobs

The command displays the CLI command used to create the job. A plus sign (+) displayed with the job indicates the most recently active job; a minus sign (-) indicates the previously active job.

kill

The **kill** command terminates a connection and ends a job. The command syntax is:

kill [*arguments*]

The **kill** command displays the job number and the CLI command that created it. The Annex accepts up to four arguments to kill multiple jobs. Valid **kill** arguments are:

none, *%*, *%%*, *%+*	Kills most recent job (+).
%-	Kills previous job (-).
n, *%n*	Kills job *n*.
%hostname	Kills job at *hostname*.

lock

The **lock** command prevents unauthorized use of the port to which the terminal is attached. The **lock** command prompts for a password, and denies access to the port until that password is entered. A **Key:** prompt appears after the port is locked, and remains until you enter the correct password. The command syntax is:

lock [*time-out*]

The **lock** command permits you to define a *time-out*. This is an amount of time in minutes during which the port is locked. After passing this limit, the Annex resets and unlocks the port (like **hangup**). Entering the Annex's administrative password, or resetting the port, also unlocks the port.

netstat

The **netstat** command displays statistics and information that the Annex has obtained from the network. The command is similar to the UNIX **netstat** command in format and display, but offers additional options. The command syntax is:

netstat [-AaimnRrSst]

The display format varies according to the options selected and the network protocols implemented for the Annex. See *Book C: Network Management, Chapter 1: Network Administration* for examples of the display format. If you enter the command without options, it displays the local and remote addresses, send and receive queue sizes (in bytes), the protocol, and the internal state of the protocol for all active connections. The command options are:

- **-A** Displays the default information along with the address of any associated protocol control blocks.
- **-a** Displays the state of all sockets, including those used by server processes.
- **-i** Displays the state of the hardware interfaces; e.g., Ethernet, SLIP, or IEEE 802.5.
- **-iS** Displays the state of the hardware interfaces plus additional information about the SLIP interfaces.
- **-m** Displays statistics for memory allocation.
- **-n** Displays all network addresses as numbers rather than names or symbols; can be used in combination with the **-A, -a, -i, -r, -t** options.
- **-R** Displays information about rotaries.
- **-r** Displays the routing tables.
- **-s** Displays network protocol statistics. (LAT statistics are displayed only if the correct **lat_key** value has been set.)
- **-sr** Displays routing statistics.
- **-t** Displays the default active connection information along with the port address.

Addresses display as either *host.port*, or *network.port*. The latter displays if a socket's address does not include a specific host address. If a host name is known, the name displays. Otherwise, the Internet address displays. The **-n** option displays the Internet and TCP ports. Unspecified or wildcard addresses and ports appear as an asterisk (*).

rlogin

The **rlogin** command connects to the specified host using the **rlogin** protocol. The command syntax is:

rlogin *host* [**-l** *user*]

The **-l** *user* argument logs you into the remote host under that *user* name.

services

The **services** command displays information about available LAT services that have been advertised by LAT nodes. This command is available only if LAT is enabled and **lat_key** is set correctly. The format and detail of this display depend on the switches and information that you supply on the command line. (The *Annex Communications Server User's Guide* provides examples of the **services** command.) The command syntax is:

services [**-v**] [*service* [*host name*]]

services [**-h**] [*host name*]

The **services** command displays a subset of the following:

Service Name	A string indicating the name of the service being offered.
Service Identification	A string indicating the service's use or purpose.
Rating	An integer typically indicating the number of resources (ports) available for the service on the indicated host.
Host Name	A string indicating the name of the host offering the service.
Host Identification	A string typically indicating the host's location or other special characteristics of the host offering the service.
Host Status	A string indicating the disposition of the host offering the service.
Facility Number	An integer indicating the facility number or the host number of the host offering the service.

Entering the **services** command alone on the command line displays a summary of available LAT services on the network. Available services are restricted by group codes (see *The group_value Parameter* in this book's Chapter 2, *na Parameters,* for more information). The summary typically includes the *service name*, *status*, and *service identification*. If multiple services have the same name, the summary includes only the service of the highest rating.

If the **services** command contains the **-v** switch, the command displays the expanded view of LAT services on the network. The expanded view includes the *service name*, *rating*, *service identification*, *host name*, *host identification*, *host status*, and *facility number* of the advertising host. If multiple services have the same name, the summary includes only the service of the highest rating.

If the **services** command contains a requested *service name*, the command displays a summary of all services having that service name, regardless of the service's rating. This summary displays the *host name* field of the requested service rather than the *service name* field.

If the **services** command contains both a requested *service name* and a *host name*, the command displays a summary of the service having that service name on the host specified by the host name.

If the **services** command contains only the **-h** switch, the command displays a summary of all available LAT services by host.

If the **services** command contains the **-h** switch and the *host name* field, the command displays a summary of all the services available from the specified host.

slip

The **slip** command converts a CLI port to a SLIP interface. This command allows users at remote hosts to dial into a modem attached to the Annex and convert the port to a SLIP interface. The command syntax is:

slip

After entering the command, the Annex displays both its Internet address and the Internet address assigned to the remote end of the SLIP interface. The remote host must be configured to use these addresses. Resetting the port returns it to CLI mode. The **help** command never displays this command at the user CLI. It displays this command only at the superuser CLI.

stats

The **stats** command displays general Annex statistics. The command syntax is:

stats [**-s**[*ports*] | [*time*]]

If you enter the **stats** command without any arguments, the statistics display includes the software version number, the type of communications server, its network address, the LAN address, and use statistics.

The **stats** command's **-s** argument displays status and statistics for all serial ports. You can enter a single port (**-s5**) or a range of ports (**-s5-10** or **-s5,7,9-12**). This argument also displays control line status in which an asserted signal appears in upper case letters and a de-asserted signal appears in lower case letters.

If you enter the **-s** argument with the *time* value, the **stats** command displays serial line statistics, pausing the number of seconds specified with *time* between each display. Pressing the **attn_char** or the **Break** key aborts the display.

stty

Using the **stty** command, which is similar to the UNIX **stty** command, you can display and change port parameters that control terminal characteristics, CLI connection options, and special characters. The command syntax is:

stty [*parameter* [*value*]]

If you enter the command without arguments, **stty** displays the current parameter settings. You can modify these parameters using the **stty** commands. Rebooting the Annex, resetting the port, or issuing a **hangup** command returns the parameters to the values assigned through **na**. See the *Annex Communications Server User's Guide* for more details.

telnet

The **telnet** command uses the Telnet protocol to connect to a host. For more details, see the *Annex Communications Server User's Guide.* The command syntax is:

telnet [[**-r**] *host* [*port*]]

If you enter the command without arguments, **telnet** enters command mode and displays the **telnet:** prompt for entering **telnet** commands. If you enter the command using the host as an argument, **telnet** opens a connection to that host. The optional *port* argument allows you to enter a TCP port number at the specified host to which **telnet** makes a connection. The **-r** switch establishes a raw connection.

The **telnet** command establishes a **telnet** connection between two ports on two machines. If the foreign port is not specified, it defaults to port 23. The local port is chosen to represent the user location as follows:

- If the user is connected to the Annex via a serial port, through a modem or a terminal, the local port is chosen as 10000 + port*100 + sequential, where *port* is the serial line number (1 to 99), and *sequential* is a number (0 to 99) used to distinguish connections, and is chosen sequentially.
- If the user is connected to the Annex via the network, by using **telnet *annexname***, the local port is chosen as 10000 + sequential, where *sequential* is a number (0 to 99) used to distinguish connections, and is chosen sequentially. If all local ports from 10000 to 10099 are in use, a random unused port in the range 20000 to 29999 is chosen.

Applications can use getpeername(2) to determine the serial port used, and select special options based on this number. For example, the host's Telnet daemon could be altered to look up the port number in a data base and use it to automatically assign terminal type and stty(1) options.

who

The **who** command displays information for the Annex ports' current users, the current users on other Annexes, and hosts running a Finger daemon. The type of information that displays for each serial and virtual CLI connection includes the user, the type of connection, the jobs the user is running, the time the connection began, any idle time, and the connection's origination. Chapter 2 in the *Annex Communications Server User's Guide* describes the **who** command in detail. The command syntax is:

who [*arguments*]

If you enter the command without arguments, **who** displays a list of all the Annex users. You can enter one or more arguments with this command. The valid arguments are:

[h=]*host*	Displays information for users with open sessions to the specified *host*. You can abbreviate host names.
[u=]*user*	Displays information for users matching the argument *user*.
[p=]*port*	Displays information for the specified *port*.
@host	Displays all users at the specified *host*.
user@host	Displays a specific *user* at the specified *host*.
-l *@host*	Displays all users at the specified *host*. A host running a Finger daemon will accept the **-l** argument.

Superuser CLI Commands

The superuser CLI commands are:

?	Displays help information for all CLI commands.
admin	Enters administrative mode.
arp	Displays and modifies the Internet-to-hardware address translation tables.
boot	Reboots the Annex.
control	Changes the state of DTR and RTS or outputs a test message.
help -m	Displays help information on macros.
hosts	Flushes the host table.
passwd	Changes the administrative password.
ping	Sends ICMP Echo Request packets to the host.
procs	Displays processes at the Annex.
su	Enters and exits superuser administrative mode.
tap	Displays input and output for a serial port.

To access the superuser CLI commands, issue the **su** command at the user CLI and then enter the Annex's administrative password. The superuser prompt has a # symbol appended to whatever CLI prompt is defined for the Annex. For example:

```
annex: su
Password:
annex#
```

Note: If you have not modified the password using **na**, i.e., the **show** command displays the password as "**< unset >**", enter the Annex's Internet address for the administrative password.

admin

The CLI **admin** command is a local substitute for the host-resident **na** command, especially in stand-alone environments. The CLI **admin** command set provides a subset of the host-resident **na** command set.

Entering the **admin** command on a superuser CLI connection puts you in administrative mode. The CLI prompt is replaced with the **admin:** prompt. The attention key or quit at the **admin:** prompt terminates the **admin** session and returns you to the CLI prompt.

The CLI **admin** command set includes:

broadcast	Sends a broadcast message to a port or ports.
help or **?**	Displays the **admin** help screen.
port	Enters the default port set.
quit	Terminates the **admin** session.
reset	Resets a port or subsystem.
show	Displays the current value of an EEPROM parameter.
set	Modifies the value of an EEPROM parameter.

These CLI **admin** commands function similarly to their **na** counterparts with the following exceptions:

- The CLI **admin** commands function on the local Annex only (there is no **annex** command).
- The **help** command provides a help summary for the CLI **admin** command set (as shown above) and does not provide help for any parameter. The **help** < **command_name** > also indicates the command's usage syntax.
- The **reset** command does ***not*** reset the connection from which the **reset** command is issued. To reset your port or connection, return to the CLI and issue the **hangup** command.
- If the **reset** command is issued without arguments, it asks if the user wishes to reset the default port set. Only one character is expected as input (< return >, **y**, or **Y** causes the default port set to be reset, and any other response terminates the command).
- Any other command issued without complete arguments responds with an error or usage message; the commands do not prompt for missing arguments.
- The **set** command does not indicate which subsystem needs to be reset. You ***must*** reset the appropriate port and/or subsystems if new parameters are to take effect immediately.

Note: Some parameters do not take effect until the Annex reboots.

- The display format of the **show** command is different. The **show** command shows the annex, port, or printer parameter values for the local Annex. The parameter information is displayed in two columns. After outputting up to 24 lines of information, if there is more information to be displayed, the **show** command displays a **more:** prompt. At this prompt, pressing **q** for quit returns to the **admin:** prompt, the attention character terminates the administration session, and any other key continues the display.
- The **port** command does not accept the lsts designator field as part of the port set value.
- The superuser can issue an **admin** command from the CLI by prefacing the command with ***admin*** (i.e., entering **admin show annex** at the CLI prompt shows the Annex parameters and then returns to the CLI).

Following is the set of host-resident **na** commands that the CLI **admin** command set does ***not*** support:

annex	The CLI **admin** command set supports only the local Annex.
boot, dumpboot	These commands exist already as separate CLI superuser commands.
copy	This command degrades in usefulness since there is no **trans-annex** copy.
echo	This feature is uselss since the **read** command is not supported.
passwd	Administrative security is controlled via the superuser (**su**) command, not by **erpcd**.
read, write	The local Annex does not have a file system to support this feature.

arp

The **arp** command displays and, optionally, modifies the Internet-to-hardware address translation table that the Address Resolution Protocol (ARP) uses. Since the Annex builds the ARP table dynamically, you rarely need to modify it. The command syntax is:

arp *hostname*
arp -a
arp -d *hostname*
arp -s *hostname hard_addr* **[temp | pub]**

The options are:

hostname	Displays the current ARP table entry for that host.
-a	Displays all entries in the table.
-d	Deletes the entry specified with *hostname*.
-s	Creates an entry for the host, specified using either a *hostname* or an Internet address, at the hardware address specified using *hard_addr*. If you do not include one of the following options, the entry is permanent and not published.
temp	The created entry is temporary and is to be deleted after 20 minutes. Temporary entries are not published.
pub	The created entry is to be published. The Annex responds to requests for the host's hardware address.

Using either the *hostname* or the **-a** option, **arp** displays a host name, if known, or a **?** in place of the host name, the Internet and Ethernet addresses, and the *time to live* (TTL) field for each entry. For example:

```
frontlobby# arp -a
thirdfloor (132.245.6.65) at 00-80-2d-00-2a-c0 ttl=20
oleom (132.245.6.12) at 00-80-20-06-34-39 ttl=19
? (132.245.6.20) at 00-80-20-01-fe-b1 ttl=19
caddy (132.245.6.25) at 00-80-2d-00-22-41 ttl=20
```

boot

The **boot** command reboots the Annex and, optionally, produces a dump of the Annex's operational code. You can set a time at which the boot is to take place. The **boot** command also sends a warning message to users attached to the Annex. The command syntax is:

boot [-adhq] [+] [*HH***:] [***MM***] [< *filename* >]**

Supported arguments are:

-a	Aborts any delayed boots that are pending.
-d	Performs a dump before rebooting.
-h	Performs a diagnostic boot using the ROM Monitor **boot** command if the Annex is in DIAG mode (see the *Annex Hardware Installation Guide*).
-q	Performs a boot without sending a warning message.

You have these options for entering the boot time:

HH:MM	The exact clock time for the boot. For example, `15:15` indicates 3:15 p.m.
+HH:MM	The number of hours and minutes before the boot takes place. For example, `+2:15` indicates a boot will take place in two hours and fifteen minutes.
+ MM	The number of minutes before the boot takes place. For example, `+15` indicates a boot will take place in fifteen minutes.

The *filename* argument identifies the name of the file in which the Annex's image is maintained. If you do not enter a *filename*, the command prompts you for a file name. Press the **Return** key at the prompt, and the Annex boots the image.

The *warning* argument allows you to enter an additional 256-character warning message. Warning messages are periodically sent out to users. If you do not specify a time delay or warning, the **boot** command generates an automatic warning message.

The following example requests a boot in one hour and fifteen minutes:

```
annex# boot +1:15
bootfile: RET
warning: Shutting down for PM
```

The Annex can request its boot file from a defined preferred load host. If that host is not defined, or does not respond, the Annex broadcasts its request and boots from the first load host to respond.

control

The **control** command is a diagnostic tool that allows you to reset DTR and RTS or to output a short test message for a specified port. The command syntax is:

control *port* [**dtr-** | **dtr+** | **rts-** | **rts+**]
control *port* **testmsg** [*times* | **forever**]

The *port* argument specifies the port; it is required.

The **dtr-** option de-asserts DTR; the **dtr+** option asserts it. The **rts-** option de-asserts RTS; the **rts+** option asserts it.

The **testmsg** option outputs a message to the port. To send a test message, the port must be configured as a CLI or, if configured as a slave port, must be opened from a user (i.e., an application on the host). After the message prompt appears, pressing the **Return** key displays the default message *The quick brown fox jumped over the lazy dogs*. You can specify the number of times the message is output using the *times* option. If you enter **forever**, the message is output until you enter a break. The following example outputs the default message ten times on Port 14:

```
annex# control 14 testmsg 10
Enter test message, or press Return for default:
: RET
```

If the port is not a CLI port, or has not been opened as a slave port from a host, the command displays *Device must be in use.*

help

In superuser mode, the **help** command supports the **-m** argument, which displays a list of all macros and their assigned *port_set* for that Annex. The command syntax is:

help –m [*name*]

For example:

```
annex01# help -m

Name    Assigned Ports          Description
============================================================
1       1-6,10-16,v             :Connect to Accounting
2       1-6,10-16,v             :Read EMAIL
3       1-6,10-16,v             :Show users on the network
4       1-6,10-16,v             :Exit system
```

If you enter **help** [*name*], the Annex displays the definition of that macro entry.

hosts

In superuser mode, the **hosts** command provides an additional argument, **-f**, which flushes entries from the host table. The command syntax is:

hosts -f [*host*]

If you enter a host name, **hosts** flushes that host from the table. Entering the command without the *host* argument flushes all entries except the Annex's own entry.

passwd

The **passwd** command changes the Annex's administrative password. The following example shows the syntax:

```
annex# passwd
Current password:
New password:
Confirm new password:
```

The Annex does not echo passwords. Pressing the **Return** key after the prompts for the new password sets the password back to its default (the Annex's Internet address).

ping

The **ping** command sends an Internet Control Message Protocol (ICMP) Echo Request message to elicit an ICMP Echo Response from a specified host. The command prints a line of output for each response returned. The command syntax is:

ping [**-r**] [**-v**] *host* [*databytes* [*count*]]

Each Echo Request includes a time stamp if the number of data bytes is greater than eight. This time stamp calculates the round-trip time and is returned unchanged in the Echo Response. The default packet size is 64 bytes, 56 of which are data and 8 are header. You can change the number of data bytes using the *databytes* argument.

The **ping** command continually sends one request per second, and displays a line of output for every response. Entering any character from the keyboard stops **ping**. The *count* allows you to send a limited number of requests. When **ping** stops, it displays a brief summary.

The options are:

- **-r** Bypasses the normal routing tables and sends the message directly to a host on an attached network. An error returns if the host is not on a directly attached network. This option can **ping** a local host through an unlisted interface in the routing tables.
- **-v** Displays the IP and ICMP packet headers for the reply from the host..

The **ping** display looks like this:

```
frontlobby# ping caddy
PING caddy: 56 data bytes
64 bytes from 132.245.6.25: icmp_seq=0. time=37. ms
64 bytes from 132.245.6.25: icmp_seq=1. time=12. ms
64 bytes from 132.245.6.25: icmp_seq=2. time=12. ms
64 bytes from 132.245.6.25: icmp_seq=3. time=12. ms
----caddy PING Statistics----
4 packets transmitted, 4 packets received, 0% packet loss
round-trip (ms) min/avg/max = 12/20/37

annex# ping caddy 64 1
PING caddy: 64 data bytes
72 bytes from 132.245.6.25: icmp_seq=0. time=16. ms
----caddy PING Statistics----
1 packets transmitted, 1 packets received, 0% packet loss
round-trip (ms) min/avg/max = 16/16/16
```

procs

The **procs** command displays information about Annex processes in a tabular format similar to that of the UNIX **ps** command. The **procs** command is used for debugging Annex software. The command syntax is:

procs [**-r**] [**-p***pid*] [**-d***dev*]

- **-r** Displays only processes that are currently running on the Annex.
- **-p***pid* Displays only processes for the specified *pid*.
- **-d***dev* Displays only processes attached to the specified *dev*. Entering a **?** displays processes without an attached device.

Following is an example of a typical display:

```
annex# procs

PID PPID SF STACK SSIZ USPTR IP CP SIG RW CTIME CPU TIME DEV NAME
  0    0 WO c84c8 1000 c9444  0  0   0 00     0 0:00.306   ? root
193    0 WO c8288  23c c8464  0  0   0 00     0 0:00.002   ? watch
194    0 SO c75c0  c3c c813c 12 12   0 00     0 0:00.034   ? route
195    0 SO c7180  43c c7550 12 12   0 00 be123 0:00.001   ? timed
196    0 SO c6d40  43c c709c 12 12   0 00 be123 0:00.000   ? erpcd
197    0 SO c33c8 203c c5320 12 12   0 00 be125 0:00.121   ? snmpd
.
.
.
233  231 Rc 98a88  63c 99044 12 12   0 00 be38a 0:00.085   ? telnet
234    0 XO 974a0  c3c 97ccc 12 12   0 00 be38b 0:00.680  v1 cli
```

Some of the fields that **procs** displays are:

PID	Process ID.
PPID	Parent process ID.
SF	Process status and creation flags. The status includes: **S** (sleeping), **W** (semaphore wait), **R** (runnable), **X** (executing – always the CLI process executing **procs**), **E** (event wait), and **Z** (zombie).
CTIME	Creation time of the process. Calculated as the number of seconds since January 1, 1970, and expressed as the last six hexadecimal digits.
CPU TIME	CPU time used by the process, in minutes, seconds, and milliseconds.
DEV	Device: a **?** for system processes, a port number for a physical port, **v** followed by a number for a virtual CLI.
NAME	The name of the process.

Table D–2 lists the Annex-specific processes.

Table D–2. Annex Processes

Process	Purpose
adm_timer	Watches serial ports for activity (idle timer, inactivity timer).
cli	One per active CLI.
erpcd	Listens for incoming ERPC requests (**na**, boot).
netdattimer	Ages the host table.
reset_mach	Listens for **na** command **reset all**.
rlogin_rdr rlogin_wtr	One pair for each active **rlogin** command.
root	The initial process.
routed	Listens for RIP messages.
telnet_cmd telnet_rdr	One pair per active **telnet** command.
telnetd_lis	Listens for incoming Telnet requests.
telnetd_rdr telnetd_wri	One pair per active incoming Telnet session.
timed	Maintains the Annex time-of-day clock.
watcher	Maintains the watchdog timer.
connect_rdr connect_wtr	One pair per active **connect** command.
fingerd_lis	User information for listener.
line_adm	Port and virtual line administrator.
lpd	Listens for **aprint** commands.
p_srvr_conv	Prompts for CLI and rotary access.
rwhod	Listens for **rwho** requests.
snmpd	Listens for **snmp** commands and requests.

su

When you enter the **su** command in superuser mode, the command returns you to the user CLI, and requires no password.

tap

The **tap** command displays input and output on a device attached to a specified port. Any input from the port performing the tap is inserted into the tapped port's input stream. The command syntax is:

tap [-aksvx] *port*

The *port* argument is the number of the port to be tapped. The options are:

- **-a** Use ANSI enhanced display mode escape sequences instead of angle brackets for highlighting all input displayed by **-k**. For printing characters, the escape sequence 033 133 061 155 octal (ESC [1 m) is output instead of <; the escape sequence 033 133 060 155 (ESC [0 m) is output instead of >. Both the escape sequences and the angle brackets are output for special characters. On many terminals, these escape sequences begin and end a graphic rendition such as bold or reverse video.
- **-k** Display input to the port. Angle brackets distinguish input from output. Additional information also appears in angle brackets:
  - Control characters: < ^J > for line feed, < ^I > for tab, etc.
  - Special characters: the characters defined as special for the tapped port, such as flow control or attention characters, display as < spcl ^S >, < spcl ^Q >, etc.
  - Line breaks: < break > and < break end >.
  - Control line state changes: < rts- > and < rts+ >, < dcd- > and < dcd+ >, etc.
- **-s** Do not display a warning message after establishing the tap. This option is useful when the device connected to the port is not a terminal, and the message might interfere with its normal operation.

-v Display output from the tapped port in verbose mode. Control codes (000 to 037 octal) display as ^*X*. DEL (177octal) display as ^?. Codes greater than 177 octal display with **M-** preceding their 7-bit representation. For example, 012, the code for line feed, displays as ^J; 212 displays as M-^J.

-x Display hexadecimal codes for all characters. Use this option with either **-k** or **-v**.

The **tap** command, when invoked without options, displays any output to the port on the terminal. Keystrokes from the terminal are interpreted as if they are typed on the port. After establishing a tap, the following message displays on the tapped port, unless the **-s** option was selected:

```
*** Warning: This port is being tapped. ***
```

To stop a tap, break back to the CLI prompt and issue the CLI **kill** for the **tap** job.

The **-k** and **-v** options allow you to use **tap** as a limited software line monitor. You can monitor traffic in both directions, including incoming special conditions (i.e., line breaks, flow control, and characters with special interpretations).

The **tap** command creates an Annex job like the **telnet** and **rlogin** commands. You can break back to the CLI prompt and execute other CLI commands. However, when **tap** is not the active job, all activity on the tapped port is suspended.

Flow control on the tapping port affects the tapped port. Suspending output on the tapping port also stops output on the tapped port.

The input displayed with the **-k** option is not guaranteed to be accurate. The order of displayed data may not match the actual time sequence of the events. All input and output data is displayed. Special characters and control line changes, however, are stored in a limited buffer; if they occur too rapidly, some may be lost.

The **who** command displays a tap on a port only when it is invoked locally in superuser mode using the CLI **su** command.

Chapter 4

Utilities

General

This chapter describes the following Annex utilities and daemons:

- **aprint**
- **ch_passwd**
- **erpcd**
- **rtelnet**

aprint

The **aprint** utility sends files directly to an Annex printer connected to either the parallel or serial line port. The **aprint** utility can be used as a direct command or integrated with the standard host print-spooling mechanism. The utility syntax is:

aprint [**-A***annex* [**-L***line*]] [**-f**] [**-F***string*] [*filename...*]

The options are:

-A*annex*	The name or Internet address of the Annex to which the printer is attached.
-L*line*	The Annex port number on which to print. If *line* is 0 or missing, the parallel port is used.
-f	Prevents **aprint** from supplying form-feeds between files. By default, **aprint** supplies a form-feed at the start and end of every file.
-F*string*	The *string* used to produce a form-feed instead of the default (^L). **-F** is ignored if **-f** is specified.
filename	The file(s) to print. If no file is specified, the standard input is used.

Note: Rev. 6.1.1 and greater of the **aprint** utility does not support any Annex operational code before Rev. 3.0.

The **aprint** utility provides the following error messages:

- **Command syntax errors:**

  ```
  Usage: aprint [-Aannex] [-L#] [-Fstring] [-f] [file]...
  ```

  Old style *-Pprinter* flag cannot be combined with new flags:

  ```
  Can't mix -A and -P flags
  Can't mix -L and -P flags
  ```

  Out of range number used for *-L#* option:

  ```
  invalid serial/parallel unit number N
  ```

- **Host system configuration error:**

  ```
  can't get service to printer
  ```

  /etc/services should have an entry like: `printer 515/tcp`

- **Initialization errors:**

  -Aannex not found on network:

  ```
  can't find host address for Annex "annexname"
  ```

  -L# port is in use or disabled:

  ```
  Annex can't access requested printer
  ```

  Old style *-Pprinter* not found in **/etc/printcap**:

  ```
  unknown printer NAME
  ```

  -Pprinter found in **/etc/printcap**, but it is not an Annex printer:

  ```
  NAME is not an Annex printer
  ```

  System network software problems occurred before connection established:

  ```
  aprint: could not get socket
  aprint: bind
  aprint: connect
  aprint: transport option (N) not available
  aprint: setsockopt (N)
  ```

- **Network errors:**

  Network error while sending form-feed before print job:

  ```
  file "filename" aborted while sending formfeed to Annex
  ```

  Network error occurred while sending contents of file:

  ```
  file "filename" aborted while sending data to Annex
  ```

  Network error occurred while sending form-feed after last file:

  ```
  error sending final formfeed to Annex
  ```

  Network errors occurred while establishing contact with Annex:

  ```
  error during Annex port select
  error during wait for ACK after port select
  ```

  Network error occurred while waiting for acknowledgement of print job completion from Annex:

  ```
  error during wait for final ACK
  ```

  Unexpected *SIGPIPE* error from system software:

  ```
  Annex connection was lost unexpectedly
  Annex connection was lost during attempt to spool "filename"
  ```

  Negative acknowledgement or timeout while waiting for Annex to acknowledge completion of print job:

  ```
  Annex didn't acknowledge final data
  ```

  System network software problems occurred during communication with Annex:

  ```
  aprint: recv/read failed
  aprint: recvmsg
  aprint: sendmsg failed
  aprint: write failed
  aprint: read failed. errno N
  aprint: rcvfrom
  aprint: sendto failed
  Unknown error
  ```

ch_passwd

The **ch_passwd** utility allows users to change their password when accessing an Annex through the Access Control Protocol (ACP) security system. This utility affects only passwords in the **acp_passwd** file. The command format is:

ch_passwd [*username*]

The **ch_passwd** command is identical to the UNIX **passwd** command in that first it prompts for the old password, and then for the new one. The superuser can change any password; other users can change only their own password.

If you change the name of the ACP password file, you must recompile both **erpcd** and the **ch_passwd** utility. The source files for both are provided with the Annex software distribution and are located in */annex_root*/**src/erpcd** in the Annex root directory. For instructions on recompiling both, see *Book B: Configuration Procedures, Chapter 7: Configuring Hosts and Servers*.

erpcd

The **erpcd**, or expedited remote procedure call daemon, responds to all Annex requests. This daemon contains two programs:

- **bfs** – the block file server (BFS) used to access host files
- **acp** – the Access Control Protocol (ACP) program for host-based security requests

When downloading operational code to Annexes, a minimum of one host, accessible to an Annex, must be running **erpcd** with the **bfs** program enabled.

A UDP port (121) for **erpcd** must be defined in the services database. See the *Annex Communications Server Installation Notes* and *Book B: Configuration Procedures, Chapter 7: Configuring Hosts and Servers*, for more details on installing **erpcd**.

The **rotaries** file defines port rotaries for any Annex on the network. This file must conform to the following conventions:

- Lines beginning with a pound sign (**#**) are comments.
- Blank lines are ignored.
- A maximum of 132 characters per line.

- Entries can be continued on the next line by preceding the new line with a backslash (\).
- The following characters have special meanings: colon (:), plus sign (+), at sign (@), slash (/), comma (,), semicolon (;), backslash (\), and pound sign (#).
- The rotary name string cannot contain a space, a tab, or a comma (,), but it can contain the syntax characters listed above if they are preceded by a backslash (\).
- Spaces and tabs can be used anywhere to improve readability.
- Spaces or tabs must delimit keywords.

The syntax for the **rotaries** file is:

rotary_name: [*keywords*] *ports@location*[*/auxport*][+ *auxaddr*]

The *rotary_name* is the name of the rotary and **must** be followed by a colon (:). The maximum length is 32 characters.

The *ports@location* specifies the serial line ports and Annexes on which the rotary resides. Specify the ports as:

- A single number. For example:
```
1@annex04

6@132.245.6.7
```
- A list of numbers separated by commas. For example:
```
1,3,6,7@annex03

12,16,28@132.24.5.6.9
```
- A ranges of numbers separated by a dash. For example:
```
6-9@annex01

2-12@132.245.6.5
```
- Any combination of the above. For example:
```
1,3,7,6-9@annex01

2-12,16,23,29@132.245.6.5
```

The Annex defined in the *location* argument must be preceded by an *at* sign (@) and can be specified using either the Internet address or a name specified in a name server database. The *location* argument supports the following two options:

@inet_addr+inet_addr	Defines an auxiliary Internet address that can also access the rotary.
@inet_addr/tcp_port	Defines a TCP port number that can also access the rotary. The number must range from 6000 through 6999.

The supported *keywords* are:

ps_visible **ps_invisible**	Controls displaying the rotary name at the port server. Only rotaries with an auxiliary Internet addresses or auxiliary TCP port numbers can be defined as invisible (**ps_invisible**). The default is **ps_visible**, except on raw rotaries which are always invisible.
telnet **raw** **binary**	Defines the protocol between the port and the device. The default is **telnet**. **raw** provides a data stream with no character processing; it is intended primarily for program access to the rotary. **binary** causes the Annex to negotiate with the host to operate in **telnet** binary mode in both directions.
direct_camp_on =	**ask** | **always** | **never** Defines how camp-on is handled. Typically, the default is **ask**; **raw** rotaries default to **never**.

The following example shows a rotary for 2400 bps modems attached to four Annexes, 132.245.6.78, 132.245.6.85 , *rabbit*, and *mouse*.

```
modem_2400: 1,2,3@132.245.6.78; 16,1,4@rabbit; \
            15@132.245.6.85; 12,1,4,5,6@mouse
```

The following example shows an Annex connected to serial ports on a host that does not have a network connection. The serial ports on the Annex at 132.245.6.38 are ports 1–8, and port 16. By assigning the auxiliary Internet address 132.245.6.104 to this rotary, users can issue the **telnet** directly to the rotary.

```
milker: 1-8,16@132.245.6.38+132.245.6.104
```

rtelnet

The **rtelnet** daemon establishes a Telnet connection between a serial line on the Annex and a character special file on the host. The **rtelnet** utility runs on top of the pseudo-terminal facility provided by UNIX hosts. A pseudo-terminal is a pair of character devices: a master side and a slave side. The slave side presents an interface resembling a tty device. The slave side is driven by the **rtelnet** process operating on the master side. The command syntax is:

rtelnet [**-bdfmrD**] *annex-name port device-name*

The supported arguments are:

-b	Causes **rtelnet** to use Telnet binary mode when communicating with the Annex.
-d	Turns on socket-level debugging.
-D	Turns on verbose-output debugging. When this argument is supplied, **rtelnet** does not fork off a daemon. Rather, **rtelnet** displays the Telnet option negotiation and all received data on the terminal on which it was started.
-f	Allows **rtelnet** to release the port when the device is no longer using it, thus releasing the port for use as a CLI. This argument is used with ports configured for the adaptive mode.
-h	Causes **rtelnet** to reset the Annex port when the connection closes.
-m	Causes the network connection to the Annex port to drop whenever the pseudo-device is closed. This argument is frequently used for outbound modems, causing them to hang up when a program like **tip** exits.
-r	Directs **rtelnet** to remove the device *device-name* if it already exists. Without **-r**, **rtelnet** exits with an error message if *device-name* already exists.
annex-name	The Annex's name or Internet address.
port	The serial port number.
device-name	The name of the character special file to be created (usually in the **/dev** directory).

You can set up **rtelnet**, including the **-r** argument, to start when the host boots. After booting, **rtelnet** provides a link between a **/dev** file on the host and an Annex port.

Index

Numbers

A

B

C

D

E

F

G

H

I

J

K

L

M

N

O

P

Q

R

S

T

U

V

W

X