Book A

Revision Level History

Revision	Description
A	Initial release.

Contents

Figures

Tables

Preface

This manual is intended for the person responsible for installation, configuration, and day-to-day administration of the Remote Annex. The Remote Annex operates in heterogeneous network environments. It can communicate with any system that supports Novell NetWare, TCP/IP, LAT, TN3270, and ARA protocols. The Remote Annex TCP/IP implementation is derived from the 4.3BSD tahoe distribution of UNIX, as are the implementations of several higher-level Internet protocols.

This manual assumes its readers have a basic familiarity with UNIX systems and network administration in general, with the host operating system, and with the serial devices connected to the Remote 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.

Supported Platforms

This document supports the Remote Annex 2000, the Remote Annex 4000, the Remote Annex 6100, the Remote Annex 5390 and the Remote Annex 5391.

Using this Manual

The manual is organized into three books:

- *Book A: Configuration Procedures* presents a general introduction to the Remote Annex as well as configuration information. It contains 15 chapters: *1: Introduction to the Remote Annex*; *2: Configuring Remote Annexes*; *3: Configuring Ports*; *4: The Port Server and Rotaries; 5: Modems*; *6: Point-to-Point Protocol (PPP)*; *7: Serial Line Internet Protocol (SLIP)*; *8: Dial-up Networking*; *9: Internet Protocol (IP)*; *10: Filtering*; *11: Internetwork Packet Exchange (IPX) Protocol*; *12: AppleTalk*; *13: Printers*; *14: Configuring Hosts and Server*s; and *15: Using Remote Annex Security.*
- *Book B: Network Management* describes using the Remote Annex's utilities to manage Remote Annexes and the network. It contains two chapters: *1: Network Administration and 2: Simple Network Management Protocol (SNMP).*
- *Book C: Reference* provides a detailed reference for the commands, parameters, utilities, and network protocols supported by the Remote Annex. This book contains six chapters: *1: na Commands*; *2: Configuration Parameters*; *3: CLI Commands*; *4: Utilities*; *5: Software Reference Guide*.

Supported Version

This manual supports Release 13.3 and above of the Remote Annex software.

Printing Conventions

This manual uses the following printing conventions:

`special type`	In examples, special type indicates system output.
`special type`	Bold special type indicates user input.
<cr>	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

Each Remote Annex hardware platform ships with the appropriate hardware guide. The remaining documentation is included with the software.

Book A

Chapter 1 Introduction to the Remote Annex

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

Since the Remote Annex was designed primarily for use with UNIX systems, its user interface looks and feels like UNIX, and the networking interface is compatible with UNIX (TCP/IP, BSD 4.2, 4.3, and 4.4).

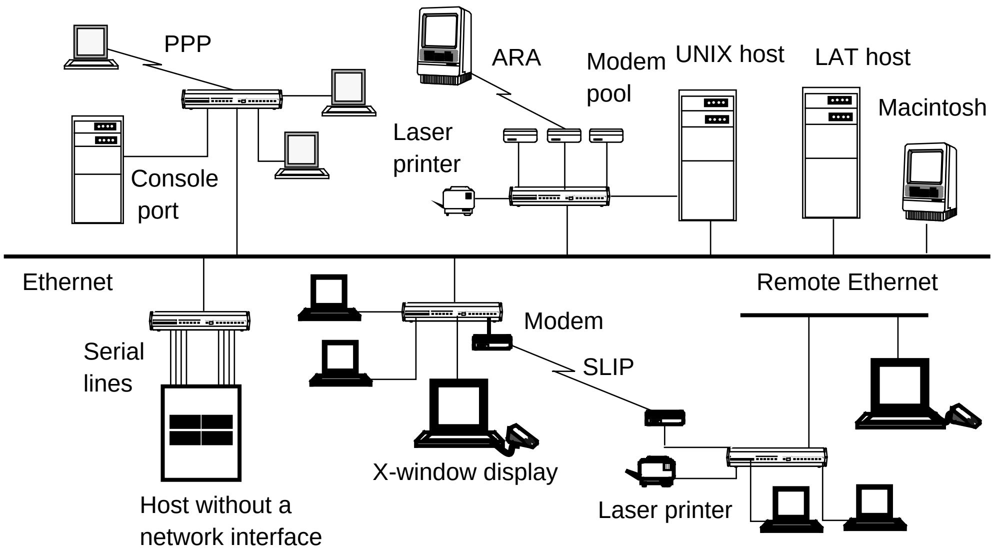


Figure A-1. Sample Local Area Network

Remote Annex Capabilities

Devices attached to the Remote Annex can easily access, or be accessed by, the network. The Remote Annex software provides network management tools and routing capabilities for managing the connections between these devices. The Remote Annex can be transparent to a UNIX host and to the application running on that host. This transparency allows devices attached to the Remote Annex to look as if they are directly connected to the UNIX host.

Network Administrator (na) Utility

The network administrator (**na**) utility is a host-based UNIX utility. It provides commands for displaying and modifying operating characteristics of the Remote 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 a Remote Annex.

Command Line Interpreter (CLI)

The Command Line Interpreter (CLI) is the Remote Annex's command interface. 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 Remote Annex and the network. The CLI also provides superuser commands for network administration and management.

The CLI **admin** command, accessed as a superuser on a CLI connection, is a local (resides in the Remote Annex) substitute for the host-resident **na** command. The **admin** command set provides a subset of the host-resident **na** commands.

Customizing the User Interface

Using the Remote 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 an alias equivalent to one or more CLI commands that connects users directly to a specific host and/or application. Or, create menus that hide the command interface, but provide the user with the appropriate selection of options.

In addition, using the Remote Annex *security profile* feature, you can customize the user environment based on user name, group name, time of day, incoming port, and protocol.

Loading Files

Remote Annex files can be loaded from a host using either the trivial file transfer protocol (**tftp**), the expedited remote procedure call daemon (**erpcd**), or the self-boot option.

- The **erpcd** utility runs on a UNIX host; it listens for Remote Annex file server host requests (download of the operational code and other files).
- The **tftp** program, supplied on most hosts, is supported as an alternative to **erpcd** and as a back-up in case a UNIX host is not available to install **erpcd**.
- The self-boot option loads and boots the operational image from local non-volatile memory. Only ROM revisions 0600 and greater with the self-boot option installed support self booting.

If a software problem occurs, the Remote Annex can send a memory dump to a host. These dumps assist Xylogics customer support personnel in resolving problems.

Extensive Security System

The Remote Annex provides comprehensive security features that assist you in securing your Remote Annexes and the network from unauthorized access. Using 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 Remote Annex. Optionally, you can use local password protection as a back-up to host-based security. You can configure the following security checkpoints:

- **CLI security**

  Access to the Remote Annex by a user at a device attached to a port
- **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 a Remote Annex
- **Virtual CLI security**

  Access to a virtual CLI connection from a host on the network
- **SLIP security**

  Access to a Serial Line Interface Protocol (SLIP) connection
- **PPP security**

  Access to a Point-to-point Protocol (PPP) connection
- **Dial-back security**

  User verification for remote connections to the Remote Annex
- **ARA security**

  Access to an AppleTalk Remote Access Protocol (ARAP) connection

For PPP, SLIP, and CLI connections, the Remote Annex also allows you to group users for security purposes and customize security based on user name, group name, time of day, incoming port, and protocol. You accomplish this by creating security profiles.

The Remote Annex provides protection through the use of an administrative password that controls access to the superuser CLI commands. This password can also protect access to a Remote Annex through **na**. The security system provides audit trails that monitor users and their activities. The Remote Annex also provides the source code for the Access Control Protocol (ACP) security system, and the flexibility to integrate Remote Annex security with existing security for a network-wide system.

The Remote Annex also supports several third-party security mechanisms, such as ACE/Server software.

Port Servers and Rotaries

The port server enables the Remote Annex to add resources to the network by allowing access to devices attached to ports. Users and applications on the network can access these devices through **rlogin** and **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 Remote Annex can manage them – as one resource. You can assign names to rotaries. Using rotaries, you can: assign multiple rotaries to one Remote Annex with each rotary having its own name; assign multiple ports on a Remote Annex to one rotary; or assign rotaries on different Remote 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 supports camp-on**

  Using camp-on, if all requested ports are busy, the port server queues the user on a first-come, first-served basis. The user can put the camp-on request into the background and resume another job; the Remote Annex signals when the port is free.

- **The port server provides access to CLI connections**

  The Remote Annex creates a virtual CLI connection when a user at the port server requests access to the CLI. The Remote Annex also provides security mechanisms for both the port server and the virtual CLI connection, including host-based user validation or local password protection, before accessing a port.

UNIX Host-originated Connections

The Remote Annex supports three utilities for UNIX host-originated connections: **lpr/lpd**, **rtelnet** and **aprint**.

Remote Annex Server Tools for Windows NT® does not support the **rtelnet** or **aprint** utilities.

The **lpr/lpd** protocol is a standard system utility for printing which is typically distributed with systems that support TCP/IP.

The **rtelnet** utility is more flexible than **aprint**; it supports many types of existing applications, and establishes **telnet** connections between a serial line on the Remote Annex and a character special file on a host. The **rtelnet** utility is a Remote 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 Remote Annex serial ports. The **rtelnet** utility allows a UNIX host to open, read, and write a pseudo-terminal corresponding to a Remote Annex port.

Using **rtelnet**, protocols such as **tip**, **cu**, **ADP**, and **kermit** can work with modems and with PCs attached to Remote Annexes. Also, **rtelnet** can be used with printing software (e.g., PostScript) that communicates bidirectionally with 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 Remote Annex's serial or parallel printer port. This utility does not provide spooling capabilities, but can be incorporated into other script files or utilities that send UNIX files to Remote Annex printers.

In general, applications written for use with the Remote Annex should not use **rtelnet** or **aprint** to connect to serial ports. Instead, the ports should be configured as slave or adaptive and applications should use TCP to connect to them directly (see *The Port Server and Rotaries* on page A-71).

Name Server Support

The Remote Annex can use 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 Remote Annex's first choice (source code for an IEN-116 name server is provided with the Remote Annex software).

A DNS server enables a local network to connect to large IP networks, such as the Internet. The Remote Annex uses a DNS server for: 1) multiple names for the same Internet address; and 2) multiple Internet addresses for the same host.

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

Network Management

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

The Remote Annex's host-based administration provides tools for downloading the Remote Annex software from a file server host. In the unlikely event of software problems, you can also obtain an Remote Annex dump.

The CLI provides commands that can duplicate the functions of a line monitor or break-out box. You can issue CLI commands remotely through virtual CLI connections.

Using the CLI commands, you can:

- Tap serial 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 a remote loopback test on other hosts.

Full Routing

The Remote Annex supports full routing that allows access to all TCP/IP hosts on the network, regardless of location. The network's complexity is not an issue: it can be simple or it can have many subnets and gateways to other networks (for more details on routing, see *IP Routing* on page A-169).

Full routing includes:

- Passive routing in which the Remote Annex uses the Routing Information Protocol (RIP) to learn routes.
- Active routing in which the Remote Annex uses RIP to advertise learned routes. (The network administrator enables this feature by setting the **option_key** parameter to a value obtained from the Remote Annex supplier).
- Hardwired routing for smaller networks.

Multi-protocol Support

The Remote Annex supports multiple protocols, allowing access to Novell NetWare, TCP/IP, ARA, and LAT hosts and services. The **telnet** and **rlogin** commands access TCP/IP hosts; the **connect** command accesses LAT services; and remote ARA users dial into the Remote Annex and become directly connected ARA network users.

Applications for the Remote Annex

The Remote Annex supports many applications that go beyond simply servicing terminals. Using a Remote Annex, you can:

- Connect terminals, X Window terminals, PCs, modems, and printers.
- Connect remote hosts, networks, and Remote Annexes.
- Connect Apple PowerBook and Macintosh computers.
- Connect hosts lacking a network interface.
- Perform remote system management.

Connecting Terminals

The Remote 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 Remote Annex provides the ability to hot-key back and forth between these sessions with user-defined key sequences. When multiple sessions have been started, those not currently in use 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.

Connecting X Window Terminals

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

Connecting PCs

By running PPP, SLIP, or CSLIP, the user can connect a PC to the network using a serial port attached to the Remote 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 is connected directly to the network.

Connecting Modems

The Remote 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 the CLI, to having restricted access to a dedicated host. The behavior of bidirectional modems is defined by whether the call comes from outside the local network through the modem or is initiated by an application or user on the LAN.

Modems attached to a Remote Annex can be grouped into a modem pool, which is easier to manage than when modems are attached to several different computers. Also, the Remote Annex's security system adds a level of protection beyond that provided by individual hosts.

Connecting Printers

Using **rtelnet**, the Remote Annex supports printers that require dynamic font downloading and bidirectional communication. The **aprint** utility supports only unidirectional printers.

The Remote Annex supports the Centronics and Dataproducts interfaces on its parallel port(s).

Connecting Apple PowerBook and Macintosh Computers using ARA

Using the Remote Annex as a remote dial-in AppleTalk Remote Access (ARA) server, Apple PowerBook and Macintosh computers can communicate with one another or with an AppleTalk network over standard telephone lines. A remote ARA user can dial into an AppleTalk network and take advantage of all available services.

Connecting Apple PowerBook and Macintosh Computers using PPP

Using the Point-to-Point Protocol (PPP), Apple PowerBook and Macintosh users can connect to a Remote Annex from a remote site and communicate with one another or with an AppleTalk network over standard telephone lines. Once connected, users can take advantage of all available services. The same PowerBook or Macintosh user can also run IP over the connection simultaneously and use IP or AppleTalk services as needed.

Connecting Hosts without a Network Interface

The Remote Annex can act as front-end to a host lacking a network interface by providing that host with an interface. By attaching the host's serial lines to the Remote Annex's serial ports, users on the network can access the host through the Remote Annex using the Telnet protocol. The Remote 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 Remote Annex supports remote system management through a connection between a computer’s console port and a Remote Annex serial port. In this configuration, you can reboot and perform kernel debugging remotely on systems to which you do not have physical access.

Book A

Chapter 2
Configuring the Remote Annex

Configuring the Remote Annex involves setting parameters to define the unit's necessary operating and administrative attributes. These administrative attributes include:

- Defining Internet addresses for the Remote Annex.
- Defining the preferred hosts for booting and dumping.
- Setting up security for the Remote 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 Remote Annex environment.
- Configuring LAT services.
- Configuring the unit for AppleTalk.

For more configuration information, see *IP Routing* on page A-169.

For more information on using the **na** commands, see *na Commands* on page C-1.

For more details on AppleTalk, see *AppleTalk* on page A-301.

Configuring Remote Annex Parameters

You can configure Remote Annex parameters using the following:

- The host-based **na** utility.
- The CLI superuser **admin** command.
- Remote Annex Manager (GUI).
- A SNMP based manager such as Sun NetManager.

To determine the current settings of Remote Annex parameters, use the **show annex all** command. 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 Remote Annex with its operational code (see the *Remote Annex Hardware Installation Guides* for more details).

By default, the **show annex** command scrolls the selected parameters line by line in two-column format.

You can set up a pager as follows:

`setenv PAGER more` (BSD)

or

`set PAGER=more; export pager` (System V)

The **./src/na/README** file describes how to use a *pager* along with the **show** command.

Using the na Utility

1. **At a terminal connected to a UNIX host, enter** na**:**

```
% na
Remote Annex network administrator Rx.x
command:
```

2. **Specify one Remote Annex, or specify multiple Remote Annexes:**

```
command: annex 192.9.200.95
         or annex 192.9.200.95,frontlobby
         or annex
enter default annex list: 192.9.200.95,frontlobby
```

3. **Execute the** set annex **command to change parameters. The following sample command lines:**

   - Enable the DNS name server.

- Define two name server hosts.
- Enable security on the Remote Annex.
- Define a 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
command: set annex pref_name1_addr 192.9.200.95
command: set annex pref_name2_addr 192.9.200.85
command: set annex enable_security Y
command: set annex vcli_security Y
command: set annex pref_secure1_host 192.9.200.95
command: set annex password piano
command: set annex syslog_mask all
command: set annex syslog_host 192.9.200.95
command: set annex cli_prompt "%a%c"
```

4. **Execute the** show annex all **command to review your changes. Using the example in Step 3, the terminal displays:**

```
command: show annex all

         Remote Annex Generic Parameters

inet_addr:132.245.44.187      subnet_mask:255.255.255.0
pref_load_addr:132.245.44.80  pref_dump_addr:132.245.33.8
load_broadcast:N             broadcast_addr:132.245.44.255
load_dump_gateway:132.245.44.22load_dump_sequence: net
image_name: "oper.46.I9336"   motd_file: "motd"
config_file: "config.annex"   authoritative_agent: Y
routed: Y                     server_capability: none
disabled_modules: vci         tftp_load_dir: ""
tftp_dump_name: ""            ipencap_type: ethernet
ip_forward_broadcast: N       tcp_keepalive: 120
option_key: "OHCg0C52T"       session_limit: 1152
output_ttl: 64
```

(continued on next page)

```
                   VCLI Parameters

max_vcli: unlimited           cli_prompt: "%a%c"
vcli_security: N              vcli_password: "<unset>"

                   Nameserver Parameters

nameserver_broadcast: N       rwhod: Y
pref_name1_addr: 192.9.200.95 name_server_1: dns
pref_name2_addr: 192.9.200.85 name_server_2: none
host_table_size: 64           min_unique_hostnames: Y

                   Security Parameters

enable_security: Y            security_broadcast: Y
pref_secure1_host:192.9.200.95pref_secure2_host: 0.0.0.0
network_turnaround: 2         loose_source_route: Y
acp_key: "<unset>"            password: "<set>"
allow_snmp_sets: N            lock_enable: Y
passwd_limit: 3               chap_auth_name: "chap"
max_chap_chall_int: 0

                   Time Parameters

time_broadcast: N             daylight_savings: us
timezone_minuteswest: 300     time_server: 0.0.0.0

                   SysLog Parameters

syslog_mask: all              syslog_facility: log_local4
syslog_host: 192.9.200.95     syslog_port: 0

                 MOP and "Login" user Parameters

pref_mop_host:00-00-00-00-00-00mop_password: "<unset>"
login_password: "<set>"        login_prompt: "#"
login_timer: 30

                   LAT Parameters

lat_key: ""                   facility_num: 0
server_name: ""               sys_location: ""
lat_queue_max: 4              service_limit: 256
keep_alive_timer: 20          circuit_timer: 8
retrans_limit: 8              group_value: none
vcli_groups: none             multicast_timer: 30
multisessions_enable: N
```

(continued on next page)

```
                 AppleTalk Parameters
a_router: 00-00-00-00-00-00
default_zone_list: ""          node_id: 0.0
zone: ""
                 Router Parameters
rip_auth: "<unset>"            rip_routers: all
                 IPX Parameters
ipx_file_server: ""            ipx_frame_type: raw802_3
ipx_dump_username: ""          ipx_dump_password:"<unset>"
ipx_dump_path: ""              ipx_do_checksum: N
                 TMux Parameters
tmux_enable: N                 tmux_max_host: 64
tmux_delay: 20                 tmux_max_mpx: 700

                 DHCP Parameters
pref_dhcp1_host                pref_dhcp2_host: 0.0.0.0
dhcp_bcast
```

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

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

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

The **write** and the **copy annex** commands do not write or copy the Remote Annex's Internet address, administrative password, virtual CLI password, LAT key, option key, or ACP key.

Using the CLI admin Command

Entering the **admin** command at a superuser CLI connection puts you in administrative mode. The *admin* prompt replaces the CLI prompt. Pressing the attention key or typing quit at the *admin* prompt terminates the **admin** session and returns you to the superuser CLI prompt (see *admin* on page C-132 for more details).

The **admin** command functions only on the local Remote Annex.

When issuing **admin** with command line arguments (not as a subsystem) you must include the *port_set*.

1. **At the CLI prompt, execute the su command:**

```
annex: su
password:
```

2. **At the superuser CLI prompt, execute the admin command:**

```
annex# admin
Remote Annex R.x.x 4 async, 0 modem ports
admin:
```

3. **Execute the set annex command to change parameters. The following sample command lines:**

   - Enable the DNS name server.
   - Define two name server hosts.
   - Enable security on the Remote Annex.

```
admin:set annex name_server_1 dns
admin:set annex pref_name1_addr 192.9.200.95
admin:set annex pref_name2_addr 192.9.200.85
admin:set annex enable_security Y
```

4. **Execute the** show annex all **command to review your changes. Using the example in step 3, the terminal displays:**

```
command: show annex all

                   Remote Annex Generic Parameters

inet_addr:132.245.44.187      subnet_mask:255.255.255.0
pref_load_addr:132.245.44.80  pref_dump_addr:132.245.33.8
load_broadcast:N             broadcast_addr:132.245.44.255
load_dump_gateway:132.245.44.22load_dump_sequence: net
image_name: "oper.46.I9336"   motd_file: "motd"
config_file: "config.annex"   authoritative_agent: Y
routed: Y                     server_capability: none
disabled_modules: vci         tftp_load_dir: ""
tftp_dump_name: ""            ipencap_type: ethernet
ip_forward_broadcast: N       tcp_keepalive: 120
option_key: "OHCg0C52T"       session_limit: 1152
output_ttl: 64

                   VCLI Parameters

max_vcli: unlimited           cli_prompt: "%a%c"
vcli_security: N              vcli_password: "<unset>"

                   Nameserver Parameters

nameserver_broadcast: N       rwhod: Y
pref_name1_addr: 192.9.200.95 name_server_1: dns
pref_name2_addr: 192.9.200.85 name_server_2: none
host_table_size: 64           min_unique_hostnames: Y

                   Security Parameters

enable_security: Y            security_broadcast: Y
pref_secure1_host:192.9.200.95pref_secure2_host: 0.0.0.0
network_turnaround: 2         loose_source_route: Y
acp_key: "<unset>"            password: "<set>"
allow_snmp_sets: N            lock_enable: Y
passwd_limit: 3               chap_auth_name: "chap"
max_chap_chall_int: 0


                   Time Parameters

time_broadcast: N             daylight_savings: us
timezone_minuteswest: 300     time_server: 0.0.0.0
```

(continued on next page)

```
                   SysLog Parameters
syslog_mask: all               syslog_facility: log_local4
syslog_host: 192.9.200.95      syslog_port: 0

                   MOP and "Login" user Parameters
pref_mop_host:00-00-00-00-00-00
mop_password: "<unset>"        login_password: "<set>"
login_prompt: "#"              login_timer: 30

LAT Parameters
lat_key: ""                    facility_num: 0
server_name: ""                sys_location: ""
lat_queue_max: 4               service_limit: 256
keep_alive_timer: 20           circuit_timer: 8
retrans_limit: 8               group_value: none
vcli_groups: none              multicast_timer: 30
multisessions_enable: N

                   AppleTalk Parameters
a_router: 00-00-00-00-00-00
default_zone_list: ""node_id: 0.0

                   Router Parameters
rip_auth: "<unset>"rip_routers: all

                   PX Parameters
ipx_file_server: ""           ipx_frame_type: raw802_3
ipx_dump_username: ""         ipx_dump_password:"<unset>"
ipx_dump_path: ""             ipx_do_checksum: N

                   TMux Parameters
tmux_enable: Ntmux_max_host:  64 tmux_delay: 20
tmux_max_mpx: 700

                   DHCP Parameters
pref_dhcp1_host               pref_dhcp2_host: 0.0.0.0
dhcp_bcast
```

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

Local File System

The stand-alone file system allows the Remote Annex to store its configuration and message-of-the-day files in local non-volatile memory. The configuration files must have the appropriate file names for the operational image to locate and load them. These files exist in the **root** directory rather than the **/usr/spool/erpcd/bfs** directory. The files are manipulated using the CLI local file system commands.

Remote Annex Internet Addressing

The Remote 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 Remote Annex's Internet address is defined in the **inet_addr** parameter. This address must be set prior to downloading the operational code to the Remote Annex. To do so, use the ROM monitor **addr** command during the Remote Annex's initial installation. You can reset the address at any time thereafter by changing the **inet_addr** parameter.

The CLI, **na**, and ROM Monitor commands always display the Internet address in dotted decimal notation.

The Broadcast Address

The broadcast address defines the Internet address the Remote Annex uses to broadcast. The Remote 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 defines this address.

The Subnet Mask

If the network is divided into subnets, you must specify the Remote Annex's Internet subnet mask using the **subnet_mask** parameter. If you do not define the subnet mask, the Remote Annex assigns one based on the network part of its Internet address. Set this parameter using the ROM Monitor **addr** command during the Remote Annex's initial installation. You can reset the address at any time thereafter by changing the Remote Annex **subnet_mask** parameter.

Certain combinations of the Remote Annex subnet mask and Internet address have special meaning:

- Setting the Remote Annex Internet address to 0.0.0.0 or 255.255.255.255 turns off all IP services, including SLIP, PPP, and IP routing. The Remote Annex continues to support non-IP services, such as ARAP and LAT, provided that they are configured properly.
- Setting the Remote Annex Internet address to a valid value and Remote Annex **subnet_mask** to 255.255.255.255 installs IP but specifies the Remote Annex does not have an Ethernet connection. IP services, including SLIP, PPP, and IP routing, are still available.

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

Booting and Dumping

The Remote Annex obtains its operational code by downloading it over the network from a UNIX host that runs Remote Annex file server software, a non-UNIX host running **tftp**, another Remote Annex configured as a boot server (running the same operational code), or the local media (self-boot). The Remote Annex boots each time it is powered up and upon receipt of a **boot** command.

The Remote Annex can dump to a file server or a host running **tftp**. The Remote Annex performs a dump upon receipt of either the **na** command **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 Remote 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 Remote Annex broadcasts its boot request and loads operational code from the first host that responds. You can modify the **pref_load_addr** parameter using **na** or the **admin** command; specify the host by its Internet address or its name.

The **image_name** parameter specifies the name of the image file that contains the Remote Annex's operational code. This file resides in different host directories, depending on which transfer protocol (**tftp** or **erpcd**) is used.

If the load host has a different network or subnet address, you must define a gateway through which the Remote Annex can reach the host. The **load_dump_gateway** parameter specifies the Internet address for the 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, enter the gateway's address using the ROM Monitor **addr** command. The Remote Annex automatically adds this gateway to its routing table (see *Creating macro Entries in the Configuration File* on page A-360 for more details).

Setting the Preferred Dump Host

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

The dump creates a file that is between one and three megabytes in size. If using **erpcd**, the Remote Annex assigns the dump file a unique name and places it in a directory named **/usr/spool/erpcd/bfs**. If using **tftp**, the file name is defined by the **tftp_dump_name** parameter and file placement is user-defined (see *Dump Host Services* on page A-403). If the dump host has a different network or subnet address, you must define a gateway through which the Remote Annex can reach the host. The **load_dump_gateway** parameter specifies the Internet address for the gateway.

Setting the Load-Dump Sequence

The configuration parameter **load_dump_sequence** specifies the network interface and the order to be used for a down-line load or an up-line dump. The arguments are **net** (for use with a LAN), **sl***nn* (for use with SLIP; *nn* is the port number), and **self** (to boot from the local media). For more details, see *load_dump_sequence* on page C-69.

Setting a Remote Annex as a Load Server

The **server_capability** parameter defines the Remote Annex as a file server host. a Remote Annex can provide operational code only for another Remote Annex of the same type. When an Remote Annex boots, it uses the **image** file to load the operational code, and the configuration file to initialize the routing table, rotaries, and macros. The Remote Annex normally does not store these files because they use memory. As a file server host, the Remote Annex uses approximately 120 Kbytes for the operational code; for the message-of-the-day (**motd**) and configuration files, it uses the amount of space relative to the size of the files.

The **server_capability** parameter defines the files that the server supplies during a boot. Table A-1 describes the arguments for **server_capability**; the default is **none**.

If you configure a Remote Annex to supply only a copy of the operational code, the default is that the Remote Annexes being booted will broadcast for the configuration and **motd** files. The file server Remote Annex uses **erpcd** to serve other Remote Annexes.

Table A-1. Arguments for the server_capability Parameter

Argument	Description
all	The Remote Annex is a file server for the configuration, operational image, and, message-of-the-day files.
config	The configuration files.
image	The operational code.
motd	The message-of-the-day file.
none	The Remote Annex is not a file server.

Disable Broadcasting for Files during a Boot

During a boot, the Remote Annex broadcasts for the configuration, **image**, and **motd** 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**.

Self Booting

The self-boot option loads and boots the operational image from local non-volatile memory. To store the image into the local media, issue the **boot –l** command from **na**, the superuser CLI, or the ROM monitor.

Only ROM revisions 0600 and greater with the self-boot option installed support the **boot –l** command

After executing a **boot –l** command, the **ls** command may not show the newly-loaded image.

To boot the stored (local) image, set the configuration parameter **load_dump_sequence** (or the ROM monitor parameter **sequence**) to **self** and reboot. This sequence instructs the Remote Annex to load the operational image and the configuration file from the local media.

To boot from both the local media and the network, set **load_dump_sequence** to either **self**,**net**, or **self, sl***xx*. The Remote Annex will first load the files from the local media; what ever files it cannot find there it will seek from the network.

For more details on the Remote Annex configuration parameters see *Configuration Parameters* on page C-33.

For more details on CLI commands, see *Using the CLI Commands* on page C-121.

Using SLIP for Booting and Dumping

You can load and dump an Remote Annex over the local area network 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 are to be used for a load or a dump and the order in which they are to be used. The Remote Annex supports booting or dumping across a SLIP interface in non-CSLIP only.

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,self,net
```

Any serial line defined using this parameter *must* be configured for SLIP (see *Serial Line Internet Protocol (SLIP)* on page A-137).

Using the Trivial File Transfer Protocol

The Trivial File Transfer Protocol (**tftp**) is a standard network interface loading program. The Remote Annex operational code opens and reads the operational image, configuration, and **motd** files. The Remote Annex accesses one file at a time.

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

The protocol used to transfer one file is independent of the protocol used to transfer another file. For environments that support both **erpcd** and **tftp**, the Remote Annex may use **tftp** to transfer one file and **erpcd** to transfer another file.

Using Remote Annex Security

The Remote 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 security, local password protection, or a combination of the two. In addition to these security mechanisms, the Remote Annex provides an administrative password that validates access through the administrative tools.

If unauthorized users can access your Remote Annex, we strongly suggest that you enable the security features after loading the host code and booting the unit.

For a detailed description of Remote Annex security, see *Using Remote Annex Security* on page A-421.

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 Remote Annex supports two standard types of name servers: a Domain Name System (DNS) server and IEN-116 server. In addition, the Remote Annex can use RWHO broadcast messages to provide name-to-Internet address translation. You can configure the Remote Annex to use one of these, a combination, or none.

The Remote 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. Host name to Internet address translation entries can be downloaded to the Remote Annex from the **gateway** section of the configuration file. The format is the same as in the **/etc/hosts** file, but aliasing is not permitted. To set up an Remote Annex for use with a name server:

- Specify the name server type.
- Specify the host(s) using the name server.
- Enable or disable the **rwhod** parameter.
- Specify the host table size.
- Enable or disable the **min_unique_hostnames** parameter.

Defining Name Servers

The Remote 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 Remote Annex allows you to specify the preferred protocol. If you choose not to use either protocol, you can configure the Remote 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 Remote 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 (see 4.3BSD documentation for more details). 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 Remote Annex to learn its name from a DNS server. The DNS' capabilities for assigning multiple aliases or multiple IP addresses to a single host allow you to assign multiple names to a rotary or multiple Remote Annexes to the same rotary (for more details, see *The Port Server and Rotaries* on page A-71).

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 Remote Annex distribution medium supplies the source for IEN-116 (see *Configuring Hosts and Servers* on page A-343 for installation instructions).

IEN-116 name servers cannot do reverse address queries.

Remote Annex Annex Server Tools for Windows NT® does not support the IEN-116 name server.

Setting Configuration 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 Remote Annex queries the name server specified with **name_server_2**. Allowable values are **dns**, **ien_116**, or **none**; the default is **none**.

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

By default, the Remote Annex does not broadcast for a name server if the preferred name servers do not respond. However, you can configure the Remote Annex to broadcast requests for a name server by setting the **nameserver_broadcast** parameter to **Y**. This causes the Remote Annex to broadcast three requests for a Domain Name Server, followed by three requests for an IEN-116 name server. You may want to use broadcast as a back-up for a name server.

Using the RWHO Protocol

Berkeley UNIX hosts use the RWHO protocol 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 Remote Annex uses the RWHO protocol as a name server. The Remote Annex runs an **rwhod** that listens for broadcasts from other hosts, but does not broadcast information about itself. When the Remote 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 satisfactory for a small network in which all the hosts run **rwhod**. But, **rwhod** often is not used in networks primarily comprised of workstations because of the load it imposes on hosts. 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 Remote 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 Remote Annex storing the wrong Internet address for that host.

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

The **rwho** parameter defines whether or not the Remote Annex listens for RWHO broadcasts. Setting the parameter to **N** disables the Remote Annex's **rwhod** and prevents the Remote 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 Remote Annex deletes the oldest, least-used entry to make room for the new one. The Remote Annex's use of the host table is erratic if the table size is too small. Increasing the size of the table reduces this problem.

You modify the host table size using 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 Remote Annex. Alternatively, you can set the size to **"none,"** which forces the Remote 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 Remote 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 Remote Annex to select *widgetslips*. Entering the name enclosed in double quotes forces the Remote 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 Remote Annex can log events to a 4.3BSD system log daemon (**syslogd**) or to a serial port on the Remote Annex. The Remote Annex may be able to log events to a 4.2BSD system using the **syslog** daemon or to a System V if it has system logging similar to 4.3BSD syslogging.

Refer to the *Remote Annex Server Tools User Guide* for *Windows NT*® for information about syslogging in a Windows NT® environment.

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.

If a system does not have a **syslog** daemon, the user can see log messages on a specific serial port by setting the Remote Annex parameter **syslog_port**. When **syslog_port** is set to a non-zero value, the Remote Annex ignores the **syslog_host** and **syslog_facility** parameters; a zero value directs the Remote Annex to use the network.

When configuring the host and the Remote Annex for system logging, consider the following parameters: **syslog_host**, **syslog_port**, **syslog_facility**, and **syslog_mask**. (Reboot the Remote Annex after configuring any parameters related to system logging.)

The **syslog_host** parameter defines the Internet address of the host configured to log Remote Annex messages. The default, **0.0.0.0**, causes the Remote Annex to broadcast its log messages.

- The **syslog_port** parameter defines the port to which syslog messages are sent. The options are **0** through the Remote Annex's port count. The default, **0**, causes the Remote Annex to log messages over the network.
- The **syslog_facility** parameter defines the facility used in the 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 in the **syslog_mask** parameter.
- The **syslog_mask** parameter defines the priority levels for logging messages. The options are **all**, **none**, or a combination of levels. The default, **none**, disables logging. Table A-2 describes the levels in priority order.

When defining a priority level, all messages of that level or greater (i.e., of greater severity) are forwarded to **syslogd**. For example, selecting **error** logs all **error**, **critical**, **alert**, and **emergency** messages.

Table A-2. Priority Levels for the syslog_mask Parameter

Level	Description
emergency	Hardware failures.
alert	All Remote Annex reboots.
critical	Configuration and initialization problems, such as format errors in the **gateway** section of the configuration 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 Remote Annex jobs created by the **rlogin** and **telnet** commands and the **ping** and **tap** superuser CLI commands.
debug	Activation and exit of all Remote Annex processes.

Using the Time Server

The Remote Annex maintains a UNIX-style time-of-day clock, which is based on the Internet date and time server. The Remote Annex distribution includes source code for a time server in case one is not available on the preferred load host. The Remote 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 Remote Annex converts time server time to local time and uses it to log events to **syslog** and to calculate the time of a boot and/or dump. The CLI **stats** and **who** commands display this time; the local file system **ls** command displays the time the files were last modified.

The Remote Annex requests the time when it boots and synchronizes its clock with a server every 30 minutes. It always queries the preferred load host first if one is defined. If a time server does not respond, the Remote Annex displays *unknown* in place of its time.

By default, if a time server is not available on the preferred load host, the Remote Annex 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 Remote Annex does not reset its time by more than 10 minutes based on an answer to a broadcast request. If the time returned to the broadcast query was greater than 10 minutes from the Remote Annex's current time, the Remote Annex only resets its time by a maximum of 10 minutes. If the timer server is on the preferred load host, the Remote 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 Remote 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 U.S. Eastern Standard Time is five hours west of GMT, its value is 300 minutes; since Paris is one hour east of GMT, its value is -60 minutes.

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

Customizing the Remote Annex Environment

You can customize the following Remote Annex attributes:

- The prompt that displays when a user accesses the CLI.
- The number of simultaneous virtual CLI connections.
- The name of the configuration file.
- The name of the message-of-the-day file.
- RIP.
- The type of IP encapsulation used by the LAN.
- TSTTY.
- TMux.
- LAT.
- AppleTalk.
- IPX.

Setting the CLI Prompt

The Remote Annex displays a prompt when a user accesses the CLI. The **cli_prompt** parameter allows you to customize the Remote Annex prompt. You can also customize the prompt for each serial port using the **prompt** port parameter (see *cli_prompt* on page C-52 and *ps_history_buffer* on page C-96).

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 specify a string for the prompt using these codes. The default is **%a%c** (*annex:*). Table A-3 describes the codes for the prompt string.

Table A-3. Formatting Codes for Remote 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 Remote 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 Remote 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.

In the following sample prompt strings a ❑ represents a space. If you want a prompt to appear as *username*❑*location*:❑, use the code:

```
%u%s%l%c
```

If the port's *username* is defined as *abercrombie* and the *location* as *lobby*, 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 Remote 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 Remote 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 Remote 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 Remote Annex.

Setting up the Configuration File

The configuration file contains all Remote Annex configuration information. It resides either on the preferred boot host or the local media and is loaded during the Remote Annex booting process (see *Configuring Hosts and Servers* on page A-343 for more details on creating and using the configuration file).

You can define a name for the configuration file using the configuration parameter **config_file**. The default file name is **config.annex**.

You can create these files on the local media using the superuser CLI **edit** command.

Setting the motd File

The Remote 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 Remote Annex through a virtual CLI connection.

The default file name is **motd**. The **motd_file** parameter allows you to specify another name for this file. The Remote Annex reads this host file each time it is booted, and when the **na** or **admin** command **reset annex motd** is issued.

Using RIP

The Remote Annex uses a routing daemon (**routed**) for its routing services. This daemon implements Versions 1 and 2 of the Routing Information Protocol (RIP).

Setting the IP Encapsulation Type

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

This parameter should be changed only at installation time using the ROM Monitor. Do not change this parameter using **na** or **admin** because the Remote Annex cannot boot with the wrong IP encapsulation.

Using the Terminal Server TTY (TSTTY)

TSTTY is a set of independent software modules that allow a host system to connect to Remote Annex serial ports in such a way that users appear as if they are directly connected to the host system. One module runs in the Remote Annex and one module runs in the host. A protocol links the two modules together. When a host wishes to talk to a device attached to a port that is in slave or adaptive mode, it must first establish a connection by connecting to the appropriate TCP port on the Remote Annex. The Remote Annex and host can then send messages over this link to exchange data and commands. (TSTTY runs on top of any reliable byte stream protocol, e.g., TCP.)

By providing a standard *tty* interface to the host, all standard programs can access the ports through standard serial port devices, and hence perform all of the functions that a standard, directly connected port can perform (for more details, see *Terminal Server TTY (TSTTY)* on page A-90).

Using the Transport Multiplexing (TMux) Protocol

The TMux protocol provides an open, standards-based solution to the CPU overload problem associated with TCP/IP terminal servers. Unlike **telnet** and **rlogin**, which generate lots of small packets, the TMux protocol multiplexes the small TCP packets generated by any number of **telnet**, **rlogin**, and TSTTY connections from an Remote Annex to a host system into a single IP network packet. Since the system load is determined per packet, not per byte, multiplexing this single packet from one system to another significantly reduces the host overhead (for more details, see *Transport Multiplexing Protocol (TMux)* on page A-97).

Configuring LAT Services

The Remote Annex can display, and connect to, currently available LAT services. Initially, all LAT functions in the Remote Annex are disabled since this feature is optional. To enable the LAT functions, the network administrator must enter the correct **lat_key** parameter value and reboot the Remote Annex (see *Configuring Hosts and Servers* on page A-343 for more details).

The **lat_key** is unique for each Remote Annex. If you purchased LAT, contact Xylogics to obtain your value.

Configuring the Remote Annex for AppleTalk

Initially, all AppleTalk functions in the Remote Annex are disabled since this feature is optional. To enable the AppleTalk functions, the network administrator must enter the correct **option_key** parameter value and reboot the Remote Annex (for more details, see *AppleTalk* on page A-301).

The **option_key** value is unique for each Remote Annex. If you purchased AppleTalk, contact your supplier to obtain a valid **option_key** value.

Configuring IPX

Initially, all IPX functions in the Remote Annex are disabled since this feature is optional. To enable the IPX functions, the network administrator must enter the correct **option_key** parameter value and reboot the Remote Annex (see *Novell Networks* on page A-271 for more details).

The **option_key** is unique for each Remote Annex. If you purchased IPX, contact your supplier to obtain a valid **option_key** value.

Chapter 3 Configuring Ports

The Remote Annex connects terminals, printers, modems, PCs, and hosts lacking a network interface to a serial line port. The Remote Annex supports several protocols that connect remote nodes to a network.

Configuring Ports

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

Using the na Utility

All port parameters have default values. When configuring ports, modify only the parameters whose defaults differ from your requirements. The **na** utility 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 either a Centronics or Dataproducts interface.

The **na** utility provides two commands for displaying the current port settings: **show port** and **show printer**. Table A-4 describes the keywords for the **show port** command.

Table A-4. Keywords for the show [port |asynchronous] Command

Keyword	Description
all	Displays all port parameters.
appletalk	Displays the port’s Appletalk parameters.
editing	Displays the CLI line editing parameters used with terminals.
flow	Displays the port’s flow control parameters.
generic	Displays the port’s basic parameters.
ipx	Displays the port’s IPX parameters.
lat	Displays the port’s LAT parameters.
ppp	Displays the port’s PPP parameters.
security	Displays the port’s security parameters.
serial	Displays the port’s serial line port parameters.
slip	Displays the port’s SLIP parameters.
timer	Displays the port’s timer parameters.
tn3270	Displays the port’s tn3270 parameters.
vci	Displays the port’s VCI parameters.

To configure a Remote Annex serial port:

1. **At a terminal connected to a UNIX host, enter** na**:**

```
% na
Annex Network Administrator Rx.x
command:
```

2. **Specify one Remote Annex or specify multiple Remote Annexes:**

```
command:annex 132.245.6.40 or
        annex 132.245.6.40,hobbes
        password:
```

3. **Specify one port or specify multiple ports:**

   command: **port 1** or **port 1–10**

   You can skip step 2 by specifying the Remote Annex with an @ following the port number(s).

   ```
   command: port 1@132.245.6.40 or
            port 1–10@132.245.6.40
   ```

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

   ```
   command:set port=8 data_bits 8 parity none autobaud N
   command:allow_broadcast Y location lab
   command:term_var vt100 dedicated_port telnet
   command:max_session_count 3
   ```

5. **Execute the** show port **command to review your changes. Using the previous example, the terminal displays:**

   ```
   command: show port=8 all

        port asy8:

        Port Generic Parameters

   mode: cli                    location: lab
   type: hardwired              term_var: vt100
   prompt: ""                   cli_interface: uci
   speed: 9600                  autobaud: N
   data_bits: 8                 stop_bits: 1
   parity: none                 max_session_count: 3
   allow_broadcast: Y           broadcast_direction: port
   imask_7bits: N               cli_imask7: Y
   ps_history_buffer: 0         banner: Y
   tcp_keepalive: 0             dedicated_address: 0.0.0.0
   dedicated_port: telnet       type_of_modem: ""
   default_session_mode:        interactive
   ```

(continued on next page)

```
          Flow Control and Signal Parameters

     control_lines: both      input_flow_control: eia
     input_start_char: ^Q     input_stop_char: ^S
     output_flow_control: eia output_start_char: ^Q
     output_stop_char: ^S     need_dsr: N
     ixany_flow_control: N    backward_key: ""
     forward_key: ""
```

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

   ```
   command: res 8
   ```

Configuring multiple ports on multiple Remote Annexes requires a few simple steps:

1. **Define a port using the** port **command.**
2. **Define the parameters for that port.**
3. **Use the** copy port **command to copy the parameters to other Remote Annex ports.**

The following example copies the parameter settings from port 1 on one Remote Annex to several ports on another Remote Annex:

```
command:annex 132.245.6.40
command:port 1
command:set port speed 9600 data_bits 8
command:copy port 1@132.245.6.40 2-8,12,14, 15,\
        16@132.245.6.55
```

You can also define all parameters for one Remote Annex including port parameters. Use the **write** command to create a script file on the specified UNIX host containing all the configuration data for that Remote Annex. Finally, execute the **read** command for all Remote Annexes you want to configure.

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

Configuring multiple ports on multiple Remote Annexes requires a few simple steps:

1. **Define a port using the** port **command.**
2. **Define the parameters for that port.**
3. **Use the** copy port **command to copy the parameters to other Remote Annex ports.**

The following example copies the parameter settings from port 1 on one Remote Annex to several ports on another Remote Annex:

```
command: annex 132.245.6.40
command: port 1
command: set port speed 9600 data_bits 8
command: copy port 1@132.245.6.40 2–8,12,14,15,\
         16@132.245.6.55
```

You can also define all parameters for one Remote Annex including port parameters. Use the **write** command to create a script file on the specified UNIX host containing all the configuration data for that Remote Annex. Finally, execute the **read** command for all Remote Annexes you want to configure.

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

Port Mode

A Remote 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 **adaptive**, **arap**, **auto_adapt**, **auto_detect**, **cli**, **connect**, **dedicated**, **ipx, ndp**, **ppp**, **rlogin**, **slave**, **slip**, **telnet**, **tn3270**, and **unused**. The Remote Annex ignores a port defined as **unused**.

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 (see *Modems* on page A-99 for details on configuring ports for modems).

A port defined as **arap** can perform as a network interface using the AppleTalk Remote Access Protocol (ARAP).

A port defined as **auto_detect** automatically determines the protocol of an incoming packet and converts to **ipx**, **ppp**, **arap**, or **cli** mode accordingly.

A port defined as **auto_adapt** automatically detects whether packets are incoming or outgoing. For incoming packets, the port behaves as if it were set to **auto_detect** mode, then determines the incoming protocol and converts to the appropriate mode. For outgoing packets, the port operates in **slave** mode.

Both **cli** and **dedicated** mean that the port is opened from the device. The difference between these two modes is that **cli** allows access to the Remote Annex’s Command Line Interface (which provides access to multiple hosts). A port defined as **dedicated** can access only one host via the connection request command (**dedicated_port** is set to **rlogin** or **telnet**). A terminal or an inbound modem typically attaches to a port defined as either **cli** or **dedicated**.

A port defined as **connect** communicates with a LAT host via the **connect** command. Use this option in conjunction with the asynchronous port parameter **dedicated_arguments**.

A port defined as **ipx** allows dial-in access via the Internet Packet Exchange protocol used by Novell Netware networks.

A port defined as **ndp** is a Novell-dedicated port; it allows dial-in and dial-out access for Novell networks, as well as routing between them.

A port defined as **ppp** using the asynchronous port **mode** parameter supports the Point-to-point Protocol.

A port defined as **rlogin** communicates via the **rlogin** command.Use this option in conjunction with the asynchronous port parameter **dedicated_arguments**.

A port defined as **slave** can be opened only by a request from the network. This access is provided through the Remote 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, and ASCII hosts lacking a network interface (for more details, see *The Port Server and Rotaries* on page A-71 and *Printers* on page A-327).

A port defined as **slip** connects any device that supports the Serial Line Internet Protocol.

A port defined as **telnet** communicates via the **telnet** command.Use this option in conjunction with the asynchronous port parameter **dedicated_arguments**.

A port defined as **tn3270** communicates via the tn3270 command. Use this option in conjunction with the asynchronous port parameter **dedicated_arguments**.

For more details on using a SLIP link, see *Serial Line Internet Protocol (SLIP)* on page A-137.

For more details on using a PPP link, see *Point-to-point Protocol (PPP)* on page A-111.

For more details on ARAP connections, see *AppleTalk* on page A-301.

For more details on Novell Netware connections, see *Internetwork Packet Exchange (IPX) Protocol* on page A-271.

Port Security

The Remote 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 (see *Using Remote Annex Security* on page A-421 for more details).

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 back-up to host-based security. It validates port access from either the device or the network. Local password protection supports **cli**, **slave**, and **adaptive** ports.

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 (for more details, see *Masking CLI Commands* on page A-554). You can customize the security policy to meet your individual security requirements. The local password protection policy cannot be altered.

To enable any security mechanism, you *must* enable security for the Remote Annex by setting the **enable_security** parameter to **Y**.

Validating CLI Connections

Host-based port security and local port password protection provide security validation on CLI connections from attached devices. The Remote 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. To use host-based security for CLI security:

- Set the port parameter **cli_security** to **Y** (see *cli_security* on page C-53).
- Create the **acp_passwd** file on the security server (see *Encrypting Security Messages* on page A-436).

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 using **port_password**. 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 Remote Annex 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 *Using the ch_passwd Utility* on page A-557 for more details).

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** (see *connect_security* on page C-54).
- Create the **acp_restrict** file on the security server (see *Using the SecurID Card* on page A-524).

The **acp_restrict** file lists those hosts and networks to which access from the specified Remote Annex is restricted. When a user requests a connection to a host, the Remote 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 a Remote Annex, all virtual CLI connections are denied access to that host.

Configuring Ports for Terminals

The Remote Annex supports several port configurations for a terminal: **cli**, **dedicated**, **slave**. The following subsections discuss how to set port parameters for the **cli**, **dedicated**, and **slave** configurations. These discussions assume that the terminal is not using a modem to connect to the Remote Annex.

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.
- Setting the **type** parameter to **hardwired** registers the user with the **who** database according to the **input_is_activity** and **output_is_activity** parameter settings. If neither parameter is set, any user on this port is invisible to **who**. If **input_is_activity** is set, when the user enters data, the line is registered with the **who** database (generally used for hardwired CLI terminals). If **output_is_activity** is set, the line is registered when the Remote Annex first sends data (generally used for hardwired printers or other slave devices). This entry is removed on a hang-up or when the slave line is released, regardless of the **input_is_activity** and **output_is_activity** parameter settings
- 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** command **broadcast** at the port.

- The **user_name** and **location** parameters are 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. The Remote Annex uses this parameter internally for the **edit** command only (see *edit* on page C-144 for more details).
- 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* on page 3-56).

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

- 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 Remote Annex 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 Remote Annex sends XOFF when it does not want to receive any more data; it sends XON when it is again ready to receive data. When the Remote Annex receives XOFF, it stops sending data to the port; when it receives XON, it resumes sending data to the port.
- An attention key, when pressed, notifies the Remote Annex that the user wishes to suspend an ongoing session with a host and return to the CLI. The Remote 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_string** defines an attention character, or string, that produces a break.
- 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

The Remote Annex allows four forms of dedicated ports for use with terminals and modems. To set a dedicated port:

- Set the serial line port parameter **mode** to **telnet**, **tn3270**, **rlogin**, or **connect** for ports using the **telnet**, **tn3270**, **rlogin**, or LAT protocols, respectively.

  In earlier software releases, setting the **mode** parameter to **dedicated** enabled access to a single host. Connection to the host was made using either the **rlogin** or **telnet** protocol. Using **telnet**, you could configure a TCP port to allow the automatic start-up of an application on the host. (Valid TCP port numbers range from 1–65535.)

  A **dedicated** port had two mechanisms for originating the connection, depending on the setting of the **control_lines** parameter. If **control_lines** was set to **modem_control** or **both**, a connection was made automatically when DCD and DSR were asserted. If **control_lines** was not set to **modem_control** or **both**, pressing the **Return** key initiated the connection.

  Ports defined as **dedicated** do not support either host-based security or local password protection.

  For backward compatibility, the Remote Annex continues to support the **dedicated** option for the **mode** parameter, but we recommend using the newer options.

- Set the **speed**, **data_bits**, **stop_bits**, and **parity** parameters to match the terminal or modem used.

- If the **dedicated** port is connected directly to a terminal, set the **type** parameter to **hardwired**. The user is registered with the **who** database according to the **input_is_activity** and **output_is_activity** parameter settings. If neither parameter is set, any user on this port is invisible to **who**. If **input_is_activity** is set, when the user enters data, the line is registered with the **who** database (generally used for hardwired CLI terminals). If **output_is_activity** is set, the line is registered when the Remote Annex first sends data (generally used for hardwired printers or other slave devices). This entry is removed on a hang-up or when the slave line is released, regardless of the **input_is_activity** and **output_is_activity** parameter settings. If the connection times out without having received input from the user, the Remote Annex prompts *Press return to restart login* and then waits for input, or a hang-up, before retrying the connection.

  If the **dedicated** port is connected to a modem, set **type** to **dial_in**. The user is registered with the **who** database as soon as a process (CLI or slave) attaches to the line, regardless of the **input_is_activity** and **output_is_activity** parameter settings. The Remote Annex continuously retries the connection, regardless of errors.

- Set the **dedicated_arguments** parameter to the target host’s name or IP address. The Remote Annex passes this string to the command given in the **mode** parameter setting, so any command line arguments, such as the optional **telnet** port number, are valid here.

Some sample settings for the **mode** and **dedicated_arguments** parameters are:

- Creating a dedicated raw TCP connection to port 7001 on another Remote Annex:

  ```
  port mode telnet
  port dedicated_arguments "-st annextwo 7001"
  ```

- Using the name-resolution interface (no hardwired IP address is required as it would be with the older **dedicated** option):

  ```
  port mode rlogin
  port dedicated_arguments "myhost"
  ```

To test the dedicated port configuration, enter the port **mode** and **dedicated_arguments** settings as a CLI command. Since the Remote Annex handles the arguments in exactly the same manner, you can easily test variations of the desired arguments until they work correctly.

When configuring a dedicated port, you may also want to set the following parameters:

- 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 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.

- If you set the **inactivity_timer** parameter, also set the **input_is_activity** or **output_is_activity** parameters (or both).
- To use EIA/hardware flow control (RTS/CTS), set the **control_lines** parameter to **flow_control** or **both**, and the **input_flow_control** and **output_flow_control** parameters to **eia**. The Remote Annex 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** or **modem_control**, and set both **input_flow_control** and **output_flow_control** to **start/stop**. The Remote Annex sends XOFF when it does not want to receive any more data; it sends XON when it is again ready to receive data. When the Remote Annex receives XOFF, it stops sending data to the port; when it receives XON, it resumes sending data to the port.
- The **telnet_escape** parameter defines the character that returns you to the telnet: prompt when using the CLI **telnet** command. Setting this parameter to **U** disables the Telnet escape. The default is **CTRL-**].

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 Remote Annex **rtelnet** utility to provide a device file for the **getty** process (see *Defining TCP Port Numbers* on page A-95 for more details).

- Set the **speed**, **data_bits**, **stop_bits**, and **parity** parameters to match the terminal's settings.

- Setting the **type** parameter to **hardwired** registers the user with the **who** database according to the **input_is_activity** and **output_is_activity** parameter settings. If neither parameter is set, any user on this port is invisible to **who**. If **input_is_activity** is set, when the user enters data, the line is registered with the **who** database (generally used for hardwired CLI terminals). If **output_is_activity** is set, the line is registered when the Remote Annex first sends data (generally used for hardwired printers or other slave devices). In any case, this entry is removed on a hang-up or when the slave line is released.
- Set the **allow_broadcast** parameter to **Y** and the **broadcast_direction** parameter to **port** so that a user at the terminal receives administrative broadcasts.
- 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 Remote Annex 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 Remote Annex sends XOFF when it does not want to receive any more data; it sends XON when it is again ready to receive data. When the Remote Annex receives XOFF, it stops sending data to the port; when it receives XON, it resumes sending data to the port.
- The CLI **who** command displays the **user_name** and **location** parameters.

Data Buffering on a Slave Port

You can buffer and display incoming data to a slave port using **telnet**. The serial port parameter **ps_history_buffer** specifies how much data to buffer. When enabled, incoming data is buffered continuously before, during, and after the **telnet** session; no data buffering occurs during LAT access to the port. After establishing a **telnet** connection to a port, the Remote Annex prompts *Display the history buffer?* only if buffered data exists. You can flush the buffer either by issuing a **telnet send ao** command or by resetting the port via **na** or **admin**.

Host-based Applications to Access a Terminal

The Remote Annex **rtelnet** daemon, which is included with the Remote 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 **rtelnet** is configured to start when the host boots, it provides the link between a **/dev** file on the host and the port. The following example sets up a **getty** line as shown in Figure A-2; the terminal is in a lobby and is connected to port 3 on *annex02*.

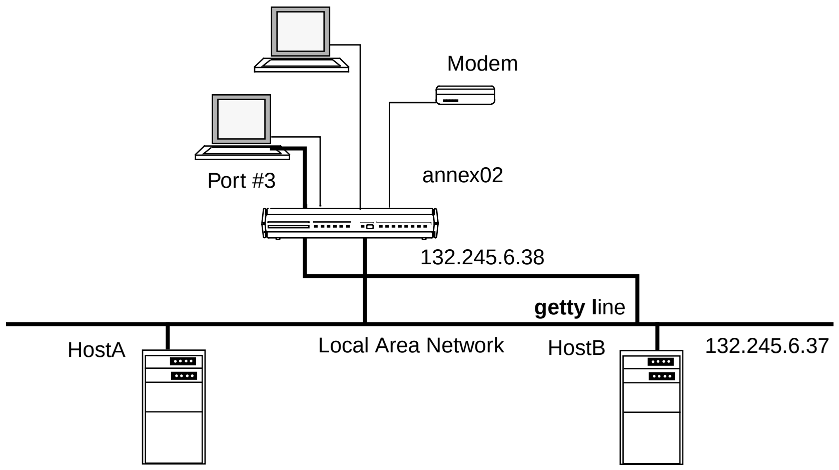


Figure A-2. Host Applications Accessing a Terminal

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

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

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

   You can specify the Remote 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/ttytab to define /dev/ttyDB as a** getty **line:**

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

3. **Signal** init **so that it reads /etc/ttytab 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 (pseudodevice) with** rtelnet **as follows:**

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

   You can specify the Remote Annex by either its IP 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 Remote Annex provides a front-end service to a host that does not have a network interface. Attach the host's serial ports to the Remote Annex's 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 Remote Annex ignores the received parity bit. Some UNIX hosts transmit different parity depending on whether **tty** is in raw or cooked mode.

- Setting the **type** parameter to **hardwired** registers the user with the **who** database according to the **input_is_activity** and **output_is_activity** parameter settings. If neither parameter is set, any user on this port is invisible to **who**. If **input_is_activity** is set, when the user enters data, the line is registered with the **who** database (generally used for hardwired CLI terminals). If **output_is_activity** is set, the line is registered when the Remote Annex first sends data (generally used for hardwired printers or other slave devices). In any case, this entry is removed on a hang-up or when the slave line is released.
- 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 Remote Annex **enable_security** parameter to **Y**.
- 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 Remote Annex 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 Remote Annex sends XOFF when it does not want to receive any more data; it sends XON when it is again ready to receive data. When the Remote Annex receives XOFF, it stops sending data to the port; when it receives XON, it resumes sending data to the port.
- The CLI **who** command displays the **user_name** and **location** parameters.

Chapter 4
The Port Server and Rotaries

As a port server, the Remote Annex accepts connection requests from users, hosts, and applications on the network (see Figure A-3). You can connect to a Remote Annex in several ways:

- Telnet
- Rlogin
- Terminal Server TTY (TSTTY)

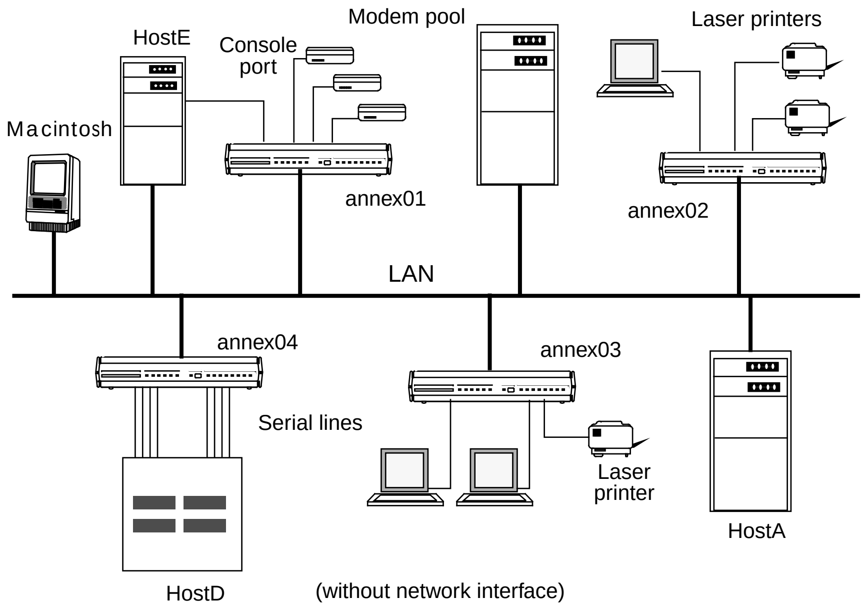


Figure A-3. Connecting Devices to a Remote Annex

The examples in this chapter use the **telnet** command. The **rlogin** command can be used in place of the **telnet** command except where noted.

Telnet and Rlogin

As a port server, the Remote Annex accepts **telnet** and **rlogin** connection requests from users and applications on the network. Once connected to the Remote 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 a Remote Annex using **telnet**:

```
% telnet annex01
Trying...
Connected to annex01.
Escape character is "^]".
Rotaries Defined:
  cli
Enter Annex port name or number: 2
Attached to port 2.
```

In Figure A-3, *annex01* provides modems for outgoing and incoming calls; it also provides the console port for remote system management on *HostE*; *annex02* and *annex03* provide laser printers to which printing applications can be sent; and *annex04* provides access to a host lacking a network interface.

The Port Server

Once connected to a Remote Annex by **telnet** or **rlogin**, the port server provides users with the option of selecting a single port or a rotary connection. **telnet** users can also select a virtual CLI connection. Applications connecting to a Remote Annex access only a single port.

A rotary is a set of Remote Annex ports grouped together so that they can be addressed by users, and managed by the Remote Annex, as a single entity. When a user requests a rotary, the port server attaches the user to the first available port. You can configure the Remote Annex to keep track of the last port to which it connected and start its port selection from that point.

Define rotaries in the **rotary** section of the configuration file. Defining rotaries allows using a name instead of a range of numbers when selecting a port. For more information on creating/using the configuration file and syntax rules for **rotary** entries, see *Configuring Hosts and Servers* on page A-343.

A virtual CLI connection permits access to CLI commands from anywhere on the network. This provides remote system management for administrators.

The **rlogin** command supports connections to defined port rotaries *only*. **rlogin** does not support connections to the conversational port server and to the virtual CLI.

When a user requests a rotary, 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 Remote Annex puts the request in a first-come, first-served queue and notifies the user when a port is free. Also, the rotary can be configured so that the user either always waits, or never waits. The Remote Annex supports ranges of port numbers that, when entered with the **telnet** or **rlogin** commands, are mapped directly to a port or to a defined rotary.

You can set up security for both the port server and the virtual CLI connections.

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 A-3 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*.

Issuing the **telnet** command at the CLI connects the user 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 Remote Annex notifies the user when the connection is complete.

In the next example, the user first presses the attention character to return to the Remote Annex prompt, then issues the CLI **bg** command to place the **telnet** request using camp-on into the background, and then issues the **fg** command to return to the *HostB* session.

The Remote 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.

```
annex: bg
2 telnet anne01 &

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

annex: fg 1
rlogin HostB
%
```

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...
```

By creating **rotary** entries in the configuration file, you can force camp-on without questioning the user or disable camp-on entirely (see *Rotaries* on page 4-81 and *Parsing the Configuration File* on page A-345).

Defining TCP Port Numbers

telnet and **rlogin** can include a TCP port number in the connection request. The Remote Annex recognizes the following ranges of TCP port numbers for **telnet** and **rlogin** connections: 5000, 6000, and 7000. (The Remote Annex recognizes TCP port numbers in the 9000 range for TSTTY connections (for more details, see *Configuring the Remote Annex for TSTTY* on page 4-94)).

A special version of **rlogin**, one that accepts TCP port numbers, is needed to use TCP port numbers with **rlogin**.

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 maps to serial port 2, TCP port 7001 maps to serial port 1, etc. (it is not necessary to configure these TCP ports).

TCP ports in the 5000 range connect directly to the specified serial port using **telnet** or **rlogin**. 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 or adaptive mode. In the next example, a user requests TCP port number 5006 using the **telnet** command:

```
% telnet annex01 5006
Trying...
Connected to annex01.
Escape character is "^]".
Attached to port 6.
```

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 results in directly attaching to the rotary (see *Rotaries* on page 4-81 for more details).

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 5900 and 7900 range map directly to parallel ports: TCP port 5901 maps to parallel port 1, TCP port 5902 maps to parallel port 2, TCP port 7901 maps to parallel port 1, and TCP port 7902 maps to parallel port 2.

The Remote Annex 4000 supports one parallel port.

Virtual CLI Connections

The Remote Annex can access the CLI from anywhere on the network through the port server. It 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 is included in the **telnet** command.

The **rlogin** command supports connections to defined port rotaries *only*. **rlogin** does not support connections to the conversational port server and to the virtual CLI.

Using a virtual CLI, you can access any CLI command. However, of all the port characteristics, only the attention character (or character string) affects the virtual CLI connection. You can define the attention character or character string using either the Remote Annex parameter **attn_string** or the **stty attn** command (see *attn_string* on page C-46 for more details). The attention character (or character string) provides access to the CLI while in a session with another host.

The Remote Annex creates a new virtual CLI connection for each request it receives. You can limit the number of virtual CLI connections the Remote Annex creates using the **max_vcli** Remote 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 back-up 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.

The following example shows the supplied user validation for port server security.

```
% 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: kate
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:

- Set the Remote Annex parameter **enable_security** to **Y**.
- Set the port parameter **port_server_security** to **Y**.
- Create a password file (**acp_passwd**) on the security server(s); see *Encrypting Security Messages* on page A-436 for more details.

For local password protection with the port server, define a password for the **port_password** parameter.

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

Security for Virtual CLI Connections

The Remote Annex establishes security for virtual CLI (VCLI) connections using host-based security, local password protection, or both. Host-based security validates the user name and user password.

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 VCLI at the port server prompt. This ensures that the user is authorized to access the VCLI.

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

- Set the **enable_security** Remote Annex parameter to **Y**.
- Set the **vcli_security** Remote Annex parameter to **Y**.
- Create a password file (**acp_passwd**) on the security server(s); see *Encrypting Security Messages* on page A-436 for more details.

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

- Set the Remote Annex parameter **enable_security** to **Y**.
- Set the Remote Annex parameter **vcli_security** to **N**.
- Define a password for all virtual CLI connections for the Remote Annex using the **vcli_password** parameter.

Local password protection can function as a back-up to host-based security:

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

Rotaries

A rotary is a group of serial ports that the Remote Annex manages as a single entity. The network administrator can customize the behavior and use of rotaries by defining **rotary** entries in the Remote Annex configuration file. The Remote Annex extracts its assigned rotary definitions from this file. Administratively defined rotaries can comprise one or more ports on one or more Remote Annexes. When customizing rotaries, you can:

- Name rotaries.
- Group together rotaries from multiple Remote Annexes.
- Assign Internet addresses.
- Assign TCP port numbers.
- Create invisible rotaries.
- Modify camp-on.
- Create raw rotaries.

- Force a binary mode connection.
- Configure port selection.

In earlier software releases, **rotaries** is an individual file. The file syntax has not changed. Using **include** statements in the configuration file incorporates files from earlier releases.

For a description of file parsing and creating **rotary** entries in the configuration file, see *Parsing the Configuration File* on page A-345 and *Creating rotary Entries in the Configuration File* on page A-381).

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 the protocol.
- Assigning phone numbers to rotaries
- Configuring port selection.

Defining Multiple Rotaries with One Entry

You can include more than one Remote Annex in a single file entry in the **rotary** section of the Remote Annex configuration file by separating the *ports@locations* field with semicolons. The following entry defines a rotary named *modems* that resides on two different Remote Annexes. The rotary on *annex01* has seven ports; the rotary on the Remote 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.
Rotaries Defined:
   modems 1,4,7,13–16
   cli
Enter Annex port name or number:
```

When the user accesses the Remote 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 Auxiliary Rotaries Internet Addresses

An auxiliary address allows you to assign an Internet address that allows a connection directly to the rotary. The user can access the rotary by entering the unique auxiliary address using the **telnet** or **rlogin** commands. 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 in the rotary.

Using **telnet**, the command line looks like this:

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

Using **rlogin**, the command line looks like this:

```
modems: 1,4,6-9@annex01+132.245.6.80/513
```

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

Users can issue the following command to access the first available port in the rotary:

```
% telnet 132.245.6.80
Trying...
Connected to 132.245.6.80.
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:

```
132.245.6.80 modems
```

Users can use the name *modems* and access the first available port in the rotary:

```
% telnet modems
Trying...
Connected to 132.245.6.80.
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 names of the rotaries. This allows users to request a rotary name using the **telnet** command.

With the DNS server, the Telnet request attempts to connect to the first IP address returned by the name server. If that connection is unsuccessful, it moves on to the next connection, and so on until a connection is available. Using the following example, one entry defines rotaries on two Remote 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 issues a **telnet** command to *modems*, 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 (assuming camp-on is set to **direct_camp_on=never**):

```
% telnet modems
Trying...
Connected to 132.245.6.80.
Escape character is "^]".
Attached to port 6.
```

When creating multiple rotaries under one name, always disable camp-on so that Telnet tries the next available port if the first connection fails.

Since a modem was not available on the first Remote Annex, **telnet** automatically crossed over to the second Remote 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** or **rlogin** commands. The last three digits of the port number are arbitrary; but the TCP port numbers must be unique for each Remote Annex.

A special version of **rlogin**, one that accepts TCP port numbers, is needed to use TCP port numbers with **rlogin**.

When users include a TCP port number in the 6000 range with the **telnet** or **rlogin** commands, 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
```

Users can issue the following command to access the first available port in the rotary:

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

Configuring Visibility

Configure visibility for rotaries using the keyword **ps=** along with the arguments **visible** and **invisible**. The port server displays a visible rotary's name when users **telnet** or **rlogin** to the Remote Annex's primary Internet address. An invisible rotary prevents users from seeing the name if they use **telnet** or **rlogin** to connect to the Remote Annex and further hides 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 1,4,6-9@annex01+132.245.6.80
```

Users that use **telnet or rlogin** to connect to *annex01* do not see the name; users that use **telnet** or **rlogin** to connect to *HostC* see the sequence illustrated in *Assigning Auxiliary Rotaries Internet Addresses* on page 4-84.

Configuring Camp-on

Camp-on applies to rotaries with an auxiliary Internet address or auxiliary TCP port. It is the process of waiting for the next free port in the rotary if all of its ports are busy when a connection is attempted. Configure camp-on using the keyword **direct_camp_on=** along with the arguments **ask**, **always**, and **never**.

The **ask** argument indicates that the port server asks the user to camp-on; **always** causes the port server to camp-on automatically; **never** causes the port server to refuse the **telnet** or **rlogin** connection if the rotary is full (**raw** rotaries default to **never**). The default is **ask**.

You must define the alternate IP address or TCP port for the rotary. Camp-on applies when using **telnet** or **rlogin** to access this alternate address or TCP port.

Configuring the Protocol

Define the protocol between the port and the device using the keyword **protocol=** along with the arguments **telnet**, **rlogin**, **tstty**, **raw**, and **binary**.

protocol=telnet

The setting **protocol=telnet** configures **telnet** as the protocol between the port and the device. This is the default setting.

protocol=rlogin

The setting **protocol=rlogin** configures **rlogin** as the protocol between the port and the device. Another way to configure rlogin as the protocol is by specifying **TCP port 513** in the **rotary** section of the configuration file.

protocol=tstty

The setting **protocol=tstty** configures **tstty** as the protocol between the port and the device (for more details, see *Configuring Rotaries for TSTTY* on page 4-96).

protocol=raw

The setting **protocol=raw** configures a raw rotary. A raw rotary passes data directly to and from the serial device -- no data processing occurs. Raw rotaries are invisible. 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.

The setting **direct_camp_on=never** is the default for raw rotaries; **ask** cannot be used. Following is an example of a raw rotary consisting of ports 1, 2, 3, and 8 on a Remote Annex whose Internet address is 132.245.6.32. The rotary is accessed through TCP port 6300:

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

protocol=binary

The setting **protocol=binary** configures a binary rotary. In this configuration, the Remote Annex negotiates with the host to operate in **telnet binary** mode in both directions:

```
strip-record: protocol=binary direct_camp_on=never 1-
3,12@132.245.6.30
```

Assigning a Phone Number to a Rotary

The keyword **phone=** defines a phone number that the Remote Annex dials automatically when a user connects to the rotary to which the number has been assigned.

Configuring Port Selection

The keyword **select=** defines the order in which the rotary selects ports. If **select=first**, the rotary selects the first available port in the *port_set*; **select=next** directs the rotary to keep track of the last port that was selected, and to start its search from that point.

In the following example, the user connects to the rotary *modems* and is attached to port 1; after disconnecting and then re-connecting to *modems*, the user is attached to port 2:

```
modems: select=next 1-5@annex01
```

```
% telnet modems
Trying...
Connected to annex01.
Escape character is "^]".
Attached to port 1.
^]
telnet> quit
```

```
% telnet modems
Trying...
Connected to annex01.
Escape character is "^]".
Attached to port 2.
```

Terminal Server TTY (TSTTY)

TSTTY is a set of independent software modules that allow a host system to connect to Remote Annex serial ports in such a way that users appear as if they are directly connected to the host system. One module runs in the Remote Annex and one module runs in the host. A protocol links the two modules together. When a host wishes to talk to a device attached to a port that is in slave or adaptive mode, it must first establish a connection by connecting to the appropriate TCP port on the Remote Annex. The Remote Annex and host can then send messages over this link to exchange data and commands. (TSTTY runs on top of any reliable byte stream protocol, e.g., TCP.)

Remote Annex Server Tools for Windows NT® does not support the **TSTTY** modules.

By providing a standard *tty* interface to the host, all standard programs can access the ports through standard serial port devices, and hence perform all of the functions that a standard, directly connected port can perform.

Many of the standard actions, such as changing the serial port's baud rate, are passed to the Remote Annex for execution, while some actions, such as dealing with a Break character, are executed by the host system. This design optimizes performance by allowing functions to be executed wherever it is best to do so. Consequently, since all standard system programs can be run on TSTTY ports, these ports can be used for login sessions (via *getty* or *ttymon*), as line printer ports using the standard spooler connections, for *uucp*/*cu* connections to other systems, and for user written programs.

TSTTY provides several advantages over **telnet** or **rlogin**:

- Process *ioctl* from a host to change the parameters of the remote port.
- Eliminates the need for special handling required by many terminal servers.
- Enables terminal servers to use standard UNIX utilities.

How TSTTY Interacts with Remote Annex Port Parameters

Since the TSTTY code on the Remote Annex receives commands from the host to change the settings of the serial port, there is a degree of unseen interaction between these settings and the default settings for the port. This interaction is similar to that of the Remote Annex CLI command **stty** and the port parameters, in that the settings are reset to their default values when the port is reset. For TSTTY, a port is reset when the host closes the last session (usually there is only one session) to the port.

Remote Annex Host Tools in a Windows NT® environment does not support the **TSTTY** modules.

The attributes for the port can be read or set using the various system calls defined for POSIX, or their UNIX equivalents. Table A-5 lists the fields from the POSIX *termios* definition that have been implemented and their interactions. Flags not listed in this table are handled by software modules on the host system, usually ldterm.

Table A-5. POSIX termios Field Definitions for TSTTY

Field	Description
c_cc	
VSTOP	Character to be used for stopping data in START/ STOP flow control mode
VSTART	Character to be used for starting data in START/ STOP flow control mode.
c_iflag	
IGNBRK	When set, the Remote Annex ignores breaks.
PARMRK	When set, characters with parity errors are returned as a multi-byte sequence.
IXOFF	When set, input flow control is set to START/ STOP or both, depending on the default setup.
IXON	When set, output flow control is set to START/ STOP or both, depending on the default setup.
IXANY	When set, any character restarts output.
ISTRIP	When set, input characters are stripped to 7 bits, otherwise all 8 bits are passed on.

(continued on next page)

Table A-5. POSIX termios Field Definitions for TSTTY (continued)

Field	Description
c_oflag	
NLDLY/ CRDLY/ TABDLY/ BSDLY/ VTDLY/ FFDLY/	The specified delay is done on the Remote Annex by waiting for all output to be sent, then starting the delay. Note: TAB3 is not a delay, and will cause host action.
c_cflag	
CBAUD	The given baud rate is used to set the Remote Annex port baud rate. If the setting is read from the Remote Annex, and there is no equivalent POSIX speed, B50 is returned. Note: split baud rates are not supported.
CSIZE	All character sizes (number of bits per character) are supported.
CSTOPB	Both 1 and 2 stop bits are supported. If this value is read from the Remote Annex, and the Remote Annex is set to 1.5 stop bits, this value is returned as 1 stop bit.
CREAD	Fully implemented.
PARENB/PARODD	Odd, even, and no parity are implemented. If the settings are read from the Remote Annex, and there is no equivalent POSIX setting, the setting is returned as no parity.
HUPCL	When set, the DTR line is dropped on close.
CLOCAL	When set, the Remote Annex will not respond to modem control changes on this line.

Configuring the Remote Annex for TSTTY

Configuring the Remote Annex for TSTTY support involves:

- Naming the TSTTY devices.
- Defining TCP port numbers.
- Defining host ports.
- Configuring rotaries for TSTTY.
- Installing the TSTTY software modules.

  The *Annex Software Installation Notes* describes how to install the TSTTY software modules.

Remote Annex Host Tools in a Windows NT® environment does not support the **TSTTY** modules.

Naming the TSTTY Devices

Since different operating systems have different requirements, the names of TSTTY devices vary from system to system. All systems have a device that is used only by the *tsttyd* program to set up the device-to-port mappings; this device is always called */dev/tstty/daemon*.

The names of the devices used to communicate with the Remote Annex ports are all numbered, starting from 0 and incrementing up to the total number of configured devices. Table A-6 lists the systems and their corresponding device names.

Table A-6. TSTTY Device Names

System Type	Device Name
System V.4	/dev/tstty/0, /dev/tstty/1, etc.
Solaris 2	/dev/term/0, /dev/term/1, etc.
SunOS/Solaris 1	/dev/tstty00, /dev/tstty01, etc.
SCO	/dev/tstty000, /dev/tstty001, etc.

Defining TCP Port Numbers

The Remote Annex port server allows the use of three types of TCP port numbers when communicating with the Remote Annex using TSTTY.

TCP ports numbers in the 6000 range can be used to select a rotary when using TSTTY. Again, the rotary must first be defined as a TSTTY rotary (for more details, see *Defining TCP Port Numbers* on page 4-76).

TCP port 9000 connects to the port server on the Remote Annex. Use this port to select a port or rotary by name, or to connect to a virtual CLI. If a rotary is selected by name, it must first be defined as a TSTTY rotary (for more details, see *Defining TCP Port Numbers* on page 4-76).

TCP port numbers in the range 9001 and up map directly to serial ports: TCP port 9001 maps to serial port 1, TCP port 9002 to serial port 2, etc.

Configuring Host Ports

Since TSTTY ports appear to the host system in exactly the same way as do directly connected ports, you can configure them in the same manner. Consult your system documentation for details. Many operating systems provide a set-up command, e.g., *sysadm* or *admintool*, that configures the ports in a way that is easy to use.

Configuring Rotaries for TSTTY

Rotaries for use with TSTTY are defined in the **rotary** section of the Remote Annex configuration file. For a rotary to specify the TSTTY protocol, you must set **protocol=tstty**. For example:

```
HostC: protocol=tstty 1,4,6-9@annex01/6300
```

It is possible to use a port in both a TSTTY rotary and a standard rotary. (for more details on defining rotaries, see *Creating rotary Entries in the Configuration File* on page A-381).

Transport Multiplexing Protocol (TMux)

The TMux protocol provides an open, standards-based solution to the CPU overload problem associated with TCP/IP terminal servers. Unlike **telnet** and **rlogin**, which generate lots of small packets, the TMux protocol multiplexes the small TCP packets generated by any number of **telnet**, **rlogin**, and TSTTY connections from a Remote Annex to a host system into a single IP network packet. Since the system load is determined per packet, not per byte, multiplexing this single packet from one system to another significantly reduces the host overhead.

Remote Annex Host Tools in a Windows NT® environment does not support **TMux**.

The *Remote Annex Software Installation Notes* describe how to install the TMux software.

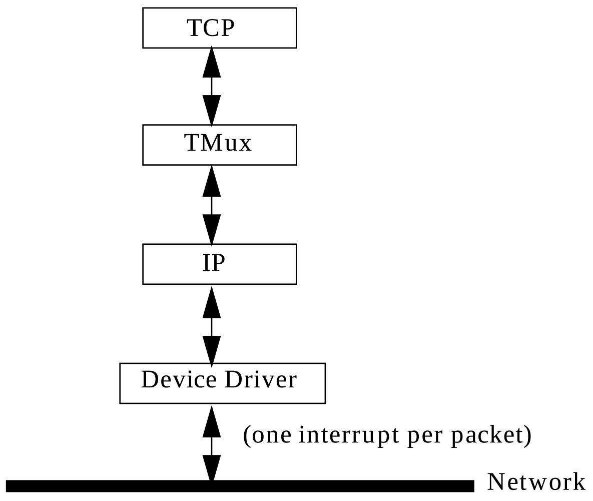


Figure A-4. How TMux Works

Each TMux packet is sent as a single IP datagram containing multiple transport segments, each preceded by a short TMux mini-header (see Figure A-5). Each of these mini-headers contains all the information required to recreate the transport segment when received by the remote end.

IP hdr	TMux hdr	Tport seg	TMux hdr	Tport seg	...

Figure A-5. TMux Packet Header

The TMux software suite consists of two independent software modules. One module runs in the Remote Annex and one module runs in the host. A protocol links the two modules together (see Figure A-6).

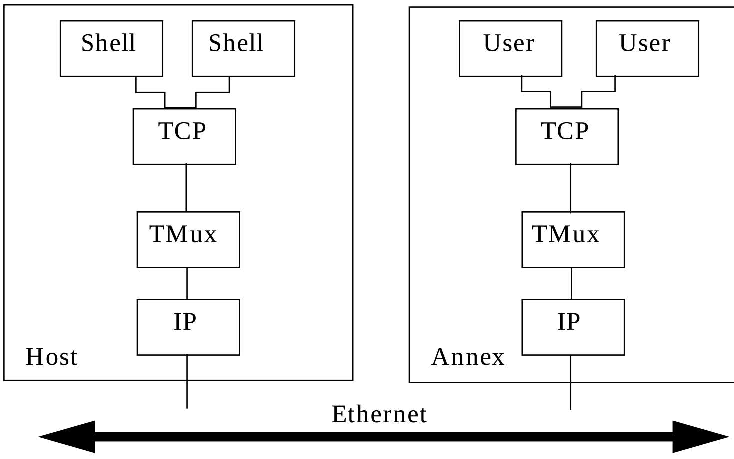


Figure A-6. TMux Block Diagram

Chapter 5 Modems

This chapter describes how to configure the Remote Annex for use with modems.

Modem Configurations

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

- Outbound: initiates only outgoing calls
- Inbound: accepts only incoming calls
- Bidirectional: accepts and initiates calls

Modem Signals

The following subsections describe the modem signals for the Remote Annex.

To successfully use US Robotics Sportster modems with the Remote Annex, set the following modem registers:

ats25=1 (tells the modem to look for a 10ms loss on DTR)
ats13=1 (tells the modem to reset on loss of DTR)

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

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 Remote Annex asserts RTS when it is ready to receive data, and checks the CTS input before transmitting data.

To use modem control (DTR/DCD/DSR), set the **control_lines** parameter to **modem_control**. The Remote Annex asserts DTR when the port is ready for use. 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 software flow control (XON/XOFF), set **control_lines** to **none**, and set both **input_flow_control** and **output_flow_control** to **start/stop**. The Remote Annex sends XOFF when it does not want to receive any more data; it sends XON when it is again ready to receive data. When the Remote Annex receives XOFF, it stops sending data to the port; when it receives XON, it resumes sending data to the port.

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 Remote Annex 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**.

When using a modem connected to a slave port, if **need_dsr** is enabled, the connection fails if no DSR signal is present. If **need_dsr** is disabled, the Remote Annex accepts the connection.

If a port with modem control enabled has **need_dsr** set to **Y**, the user cannot make a connection until DSR is active, but then can communicate out the port even if DCD is not asserted.

Setting Remote Annex Port Configuration Parameters for Modems

The following subsections discuss how to set port parameters for use with modems.

Outbound Modems

- Set the **mode** parameter to **slave**.
- Define the type of modem that is connected to the port via the **type_of_modem** parameter. This 16-byte string should match the definition of the *type_of_modem* field in the **modem** section of the Remote Annex configuration file.
- Setting the **type** parameter to **dial_in** registers the user with the **who** database as soon as the slave process attaches to the line.
- Set the **speed** parameter to the speed of the modem. Do not use **autobaud**.
- Set the **control_lines, input_flow_control,** and **output_flow_control** parameters as desired (see *Modem Signals* on page 5-99 for more 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**.
- 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 Remote Annex accepts the connection.
- 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.
- 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 Remote Annex drops DTR, terminates all sessions to host(s), and resets the port. The default setting is **off**. Set the **output_is_activity** parameter to **Y**.
- If desired, 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.

In the following example, port 13 is configured for a 2400 bps outbound modem:

```
command:set port=13
command:set port speed 9600
command:set port control_lines modem_control
command:set port type dial_in
command:set port mode slave
command:set port type_of_modem T2500
command:set port inactivity_timer 15
command:set port broadcast_direction network
command:set port input_flow_control start/stop
command:set port output_flow_control start/stop
```

Inbound Modems

Enable the modem’s *quiet* or *quiet on answer* mode (e.g., Q1 or Q2 for a Hayes) so that it does not send result codes to the port.

- You can set the **mode** parameter to either **cli** or **dedicated**. Setting **mode** to **cli** allows the user access to the command line interface. Setting **mode** to **dedicated** provides a connection to the designated host.

  If **mode** is set to **dedicated**, define the host to which this connection is made using the **dedicated_arguments** parameter. Set the **dedicated_port** parameter to either **telnet** or **rlogin** (the default is **telnet**).
- Define the type of modem that is connected to the port via the **type_of_modem** parameter. This 16-byte string should match the definition of the *type_of_modem* field in the **modem** section of the Remote Annex configuration file.
- Setting the **type** parameter to **dial_in** registers the user with the **who** database as soon as the CLI process attaches to the line. For dedicated ports, the Remote Annex continuously retries the connection, regardless of errors.
- Set the **speed** parameter to the speed of the modem.
- Set the **control_lines, input_flow_control,** and **output_flow_control** parameters as desired (see *Modem Signals* on page 5-99 for more details).
- The default setting for software flow control is **start/stop** (XON/XOFF). This setting may not be suitable for file transfer applications (**uucp**, **kermit**, **xmodem**). Do not set software flow control to **start/stop** if the port is configured for use with SLIP. PPP can use **start/stop** if the **ppp_acm** parameter is set to escape the **start** and **stop** characters.
- 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**.
- Setting the **cli_inactivity** parameter causes the port to reset 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 Remote Annex drops DTR, terminates all sessions to host(s), and resets the port. The default is **off**. Set the **input_is_activity** parameter to **Y**.
- If the port **mode** is set to either **cli** or **dedicated**, the typical setting for the **term_var** parameter is **dialup**.
- To enable CLI security, set the **cli_security** and **connect_security** parameters. To enable port access via a password, set the **port_password** parameter.
- Set the **attn_string** parameter to define a control character or character string. Setting this parameter when using a modem is helpful because modems do not always respond to the **Break** key by signalling the Remote Annex to suspend the session with the host and return to the CLI prompt. However, set **attn_string** to **off** when using file transfer programs (**uucp**, **kermit**, **xmodem**); otherwise the remote user must issue the CLI **stty** command to turn off the parameter.

In the following example, a CLI port is configured for a 2400 bps inbound modem:

```
command:set port=12
command:set port speed 2400
command:set port control_lines modem_control
command:set port type_of_modem T2500
command:set port type dial_in
command:set port mode cli
command:set port inactivity_timer 15
command:set port term_var dialup
command:set port cli_security y
command:set port cli_inactivity 10
```

Bidirectional Modems

Enable the modem’s *quiet* or *quiet on answer* mode (e.g., Q1 or Q2 for a Hayes) so that it does not send result codes to the port.

- Set the **mode** parameter to **adaptive**.
- Define the type of modem that is connected to the port via the **type_of_modem** parameter. This 16-byte string should match the definition of the *type_of_modem* field in the **modem** section of the Remote Annex configuration file.
- To enable CLI security for a port configured for incoming calls, set the **cli_security** and **connect_security** parameters. To enable port access via a password, set the **port_password** parameter.
- Setting the **type** parameter to **dial_in** registers the user with the **who** database as soon as the CLI process attaches to the line.
- Set the **speed** parameter to the speed of the modem.
- Set the **control_lines, input_flow_control,** and **output_flow_control** parameters as desired (see *Modem Signals* on page 5-99 for more details).

- The default setting for software flow control is **start/stop** (XON/XOFF). This setting may not be suitable for file transfer applications (**uucp**, **kermit**, **xmodem**). Optionally, remote users can issue the CLI **stty** to turn off flow control.

  Do not set software flow control to **start/stop** if the port is configured for use with SLIP. PPP can use **start/stop** if the **ppp_acm** parameter is set to escape the **start** and **stop** characters.
- When using a modem connected to a bidirectional port, if the **need_dsr** parameter is enabled, the connection fails if no DSR signal is present; if **need_dsr** is disabled, the Remote Annex accepts the connection.
- Set the **data_bits**, **stop_bits**, and **parity** to match the modem's requirements.
- Setting the **cli_inactivity** parameter causes the port to reset 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 Remote Annex drops DTR, terminates all sessions to host(s), and resets the port. The default is **off**. Set both **input_is_activity** and **output_is_activity** to **Y**.
- Set the **attn_string** parameter to define a control character or character string. Setting this parameter when using a modem is helpful because modems do not always respond to the **Break** key by signalling the Remote Annex to suspend the session with the host and return to the CLI prompt. However, set **attn_string** to **off** when using file transfer programs (**uucp**, **kermit**, **xmodem**); otherwise the remote user must issue the CLI **stty** command to turn off the parameter.

In the following example, a port is configured for a 9600-baud bidirectional modem:

```
command: set port=16
command: set port speed 9600
command: set port control_lines both
command: set port type_of_modem T2500
command: set port type dial_in
command: set port mode adaptive
command: set port output_is_activity y
command: set port inactivity_timer 20
command: set port input_flow_control eia
command: set port output_flow_control eia
```

Setting Up Applications

The following subsections provide set-up procedures that allow applications to access the Remote Annex port to which a modem is attached. These procedures use the **rtelnet** utility (for more details, see *rtelnet* on page C-244). All ports to which **rtelnet** attaches must be in adaptive or auto_adapt mode. Set the port parameter **mode** to either **adaptive** or **auto_adapt**.

Remote Annex Server Tools for Windows NT® does not support the **rtelnet** utility. As a result, you cannot use rtelnet-applications such as **tip** or **uucp**.

tip and uucp

The following procedure sets up applications, such as **tip** and **uucp**, to access a modem:

1. **Use the** rtelnet **utility to create a special file that references a Remote Annex port. Using** rtelnet **allows setting up the application as if** /dev/modem **is directly connected. The following example ties** /dev/modem **to port 13 on *annex02*.**

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

In the previous example, the **–m** argument instructs the Remote Annex to drop momentarily the network connection to the Remote Annex port when the pseudo device is closed; it also causes the modem to hang up when the application exits. The **–r** argument directs **rtelnet** to remove the *device name* if it already exists; without **–r**, **rtelnet** exits with an error message if the *device name* already exists.

2. **Add** rtelnet **to** /etc/rc **(or the appropriate file) 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.**

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

To access a modem attached to an **adaptive** or **auto_adapt** port:

1. **Use the** rtelnet **utility to create a special file that references a Remote Annex port. Using** rtelnet **allows setting up the application as if** /dev/modem **is directly connected. The following example ties**
   /dev/modem **to port 13 on the Remote Annex named *annex02*.**

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

In this example, the **–f** argument instructs the Remote 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 instructs the Remote Annex to drop momentarily the network connection to the Remote Annex port when the pseudo device is closed; it also causes the modem to hang up when the application exits. The **–r** argument directs **rtelnet** to remove the *device name* if it already exists; without **–r**, **rtelnet** exits with an error message if the *device name* already exists.

2. **Add** rtelnet **to** /etc/rc **(or the appropriate file) so that the special file is created when the system 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.**

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

getty

The following procedure sets up an application, such as **getty**, to access a modem:

1. **Use** rtelnet **to create a special file that references a Remote Annex port:**

```
# rtelnet -mr annex02 3 /dev/modem
```

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:

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

   Then, signal **init** to read the file and start a **getty** for that port:

```
# kill -HUP 1
```

Add **rtelnet** to **/etc/rc** (or the appropriate file) so that the special file is created when the system boots.

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

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

Chapter 6 Point-to-point Protocol (PPP)

This chapter describes the Remote Annex's implementation of the standardized method for transmitting datagrams from multiple protocols over serial point-to-point links. PPP provides three functions:

- Asynchronous High-level Data Link Control (HDLC) to encapsulate the packets.
- Link Control Protocol (LCP) to establish the connection between peers.
- A family of Network Control Protocols (NCPs) to configure network interfaces.

PPP allows a site to use the Remote Annex as a wide area network (WAN) hub, tying distant corners of an IP network together over low cost phone lines. PPP features include:

- FCS error checking.
- Agreement by each end of the connection on a mutually acceptable set of features for that connection.
- Large Maximum Receive Unit (MRU) size -- negotiations start at 1500.

In setting up a PPP link:

- You can attach a PC to a Remote Annex serial port. Using PPP as the network interface, the PC becomes a host on the network. For a remote PC with a PPP client that supports scripting, you can configure a port in CLI mode. Then, a user at a remote PC can dial into the Remote Annex using a modem and convert the port from a CLI to a PPP link using the CLI **ppp** command. After converting the port to a PPP link, the remote PC becomes a host directly attached to the network.

- Alternatively, you can set a Remote Annex port to **auto_detect** or **auto_adapt**. When the Remote Annex detects the PPP protocol on the line attached to the port, it automatically converts the port mode to PPP and starts LCP negotiations.
- If your PPP client is expecting to connect to a server already in **ppp** mode, the Remote Annex port's mode *must* be **ppp**, **auto_detect**, or **auto_adapt**.
- You can use PPP to connect two separate networks, routing data from one network to the other over the PPP link.

The Remote Annex cannot boot over a PPP link.

PPP Configuration Overview

To configure the Remote Annex for PPP sessions:

1. **Decide how IP addressing will be handled.**
2. **Review the default port parameters, then reset the parameters you need for the PPP configuration.**

Step 1: Decide How to Assign IP Addresses

All IP addressing for PPP links is based on the value of the **address_origin** parameter, which determines the method that the Remote Annex uses to assign IP addresses. The addressing methods and their corresponding **address_origin** values are as follows:

- Setting the **address_origin** parameter to **dhcp**. This enables dynamic IP addressing, using the Dynamic Host Configuration Protocol (DHCP). Refer to *Dynamic Allocation of Network Addresses* on page A-479 for a complete description of dynamic addressing.

- *Using the **acp_dialup** file* (for more details, see *Creating the acp_dialup File* on page A-481). Setting the **address_origin** parameter to **acp** causes the IP addressing for individual users to be determined by the **acp_dialup** file. This method may also be used to enable dynamic IP addressing via DHCP.

  In this book, this method is also referred to as *dial-up addressing*.

- Using the asynchronous port parameters **local_address** and **remote_address**. Setting the **address_origin** parameter to **local** (the default) causes IP addresses to be assigned according to the values of the port parameters (for more details, see *Configuration Parameters* on page C-33).

  In this book, this method is also referred to as *fixed addressing*.

You can choose to configure the Remote Annex for any one of the methods, but setting **address_origin** to DHCP has priority over addressing using the **acp_dialup** file, which has priority over addressing using the asynchronous port parameters. For information about how the Remote Annex operates when both dial-up and fixed addressing are enabled, see *Configuration Parameters* on page C-33.

Setting address_origin to dhcp

Setting the **address_origin** parameter to **dhcp** enables dynamic IP addressing. Dynamic IP addressing eliminates the need to assign remote client IP addresses manually (and the subsequent need to reconfigure and reboot) each time a host is added or moved to a new subnet. Addresses assigned in this way are used only as long as the remote client connection is active.

When DHCP is enabled, the RA 6300 acts as a DHCP client-by-proxy, requesting a remote PPP client address first from the DHCP server specified by the **pref_dhcp1_addr** parameter, then, if that server does not respond, from the DHCP server specified by the **pref_dhcp2_addr** parameter. For more information on DHCP, see *Dynamic Allocation of Network Addresses* on page A-479.

About Addressing Via the acp_dialup File

Addressing using the **acp_dialup** file offers the ability to assign IP addresses to individual users. When the **address_origin** parameter is set to **acp**, the Remote Annex uses the host-resident **acp_dialup** file to handle IP addressing. The file resides in the Remote Annex install directory. For information on making entries into the **acp_dialup** file, see *Creating the acp_dialup File* on page A-481.

Any ACP address request that comes from the Remote Annex includes the Remote Annex address and an associated user name, which are used as keys in this file. Once the keys are matched, the corresponding user addresses are returned to the caller on the Remote Annex.

You can also use the **acp_dialup** file to enable dynamic IP addressing for individual users by setting the remote address field of the file to **dhcp (**for more details, see *Dynamic Allocation of Network Addresses* on page A-479).

About Addressing Using Asynchronous Port Parameters

Setting the **address_origin** parameter to **local** causes IP addressing for the Remote Annex to be controlled by the values of two asynchronous port parameters, **local_address** and **remote_address**. This method of fixed IP addressing associates IP addresses with specific ports. For complete information on the use of the **local_address** and **remote_address** parameters, refer to *Determining Dial-up Addresses using the acp_dialup File* on page A-482.

Step 2: Review and Reset Port Parameters

The Remote Annex ships with a set of default port parameters already stored in non-volatile RAM. Review the defaults to determine which ones you need to change to satisfy your configuration requirements for PPP, security, etc.

The remainder of this section provides the following information:

- A list of the default settings for the Serial Networking and PPP port parameter groups.
- Instructions for changing a port parameter setting. Instructions for using the **set pri b** command to associate IP addresses with Remote Annex PRI B channels.

To view the entire set of default port parameters use **na** or **admin** to issue the **show port all** command.

Default PPP-Related Port Parameters

Table A-7 lists the default parameters related to the PPP protocol stored in the Remote Annex nonvolatile memory when shipped. You can view these PPP-specific parameters through the **show port ppp** command issued from the **na** or **admin** utility.

Table A-7. Default PPP-related Port Parameters Settings

Parameter	Default Setting
local_address	0.0.0.0
metric	1
net_inactivity	off

(continued on next page)

Table A-7. Default PPP-related Port Parameters Settings (continued)

Parameter	Default Setting
allow_compression	N
address_origin	local
slip_ppp_security	N
do_compression	N
net_inactivity_units	minutes
ppp_mru	1500
ppp_security_protocol	none
ppp_password_remote	"<unset>"
ppp_ipx_network	00000000
ppp_ipx_node	00-00-00-00-00-00
ppp_acm	0x0
ppp_username_remote	““
ppp_ncp	all

How to Change a Port Parameter Setting

To change a port parameter setting using **na**:

1. **At a terminal connected to a UNIX host, enter:**

   ```
   % na
   ```

   The following prompt displays on the screen:

   ```
   Annex network administrator Rx.x November 1996
   COMMAND:
   ```

2. **Specify the Remote Annex on which you intend to change port parameter settings at the** COMMAND: **prompt. Specify the administrative password for host at the** password: **prompt.**

   You can specify the Remote Annex by its IP addresses or name. If you intend to change port parameter settings on more than one Remote Annex, separate their IP addresses or names using a comma (,). If prompted for a password, the password is the administrative password for the host on which **na** is running.

   For example:

```
COMMAND:annex 132.245.6.40 or
        annex 132.245.6.40,132.245.6.45
        password:
```

3. **Specify a new setting for the port parameter at the** COMMAND: **prompt.**

   For example, to change the default setting of the **address_origin** parameter (**local**) to enable IP addressing through the **acp_dialup** file, enter the following:

```
COMMAND:set port address_origin acp
```

   The new parameter setting is stored automatically in non-volatile RAM.

4. **To review your changes, issue the** show port all **command at the** COMMAND: **prompt.**

   This command displays all of the port parameter settings. To locate the parameters you changed, press the **Return** key which allows you to scroll down through the file.

```
COMMAND: show port all
```

5. **Enter** quit **at the** COMMAND: **prompt to exit** na**.**

```
COMMAND: quit
```

Using the CLI ppp Command

The **ppp** command defines the port as a PPP network interface. This command enables a modem port to support both terminal users and network devices such as work stations and X-terminals. When the port is reset, it reverts to its original **mode**: **cli**, **adaptive**, **auto_detect**, or **auto_adapt**.

After you enter the command, the Remote Annex displays: *switching to PPP, starting LCP negotiations.* Although the **ppp** command is a user level command, it is not displayed by the **help** command. The superuser **help** command displays **ppp**.

If you start a PPP connection by issuing the **ppp** command, the PPP client software on your PC must support scripting.

Using the CLI netstat Command

The **netstat –ip** *device_id* command displays configuration and statistical data for serial interfaces. The *device_id* argument specifies a serial port.

Displaying Data for Ports

Ports are specified by port number alone, or the string *asy* followed by the port number (with no intervening white space). Each of the following sample commands specify PPP port 1:

```
netstat -ip 1
```

or

```
netstat -ip asy1
```

Creating the acp_dialup File

The **acp_dialup** file resides in the install directory, for which the default is **/usr/annex**. Any ACP dial-up address request that comes from the Remote Annex has a user name, a port, and an IP address which are used as keys in this file. If the keys match, the corresponding dial-up addresses are returned to the caller on the Remote Annex (PPP link). If no match is found, the request is denied. For a complete description of the **acp_dialup** file, refer to *Creating the acp_dialup File* on page A-481.

Dial-up addresses can also be assigned dynamically using DHCP. Refer to *Dynamic Allocation of Network Addresses* on page A-479 for a complete explanation.

Routing across a PPP Link (Passive RIP)

Both active and passive routing are available via the Routing Information Protocol (RIP) on the Remote Annex. The following sections deal with using only the most basic features of passive RIP and are intended for administrators who need minimal routing features. For complete information on both passive and active RIP, see *IP Routing* on page A-169.

Both active and passive RIP are enabled by default. To turn off active RIP, set the interface parameter **rip_advertise** to **none** for all Remote Annex interfaces (see *rip_advertise* on page A-222).

The Remote Annex bases its routing table on the information you specify in the **gateway** section of the configuration file. As a passive gateway, the Remote Annex updates the table according to RIP information it receives from other routers, but does not broadcast routing information itself, as an active gateway would. This means that the Remote Annex with the PPP interface forwards packets addressed to the host at the remote end of the connection, but does not inform other hosts or Remote Annexes that it has this capability. Other hosts, routers, and Remote Annexes on the same network must be told about the route before they can use it.

Route Cache

The route cache is a list of routing entries stored by the Remote Annex. When the Remote Annex boots, the route cache is created from the **annex...end** and **subnet...end** blocks in the **gateway** section of the configuration file (see *Creating gateway Entries in the Configuration File* on page A-350 for more details). When **routed** starts, entries in the route cache are added to the routing table if their next hop addresses (i.e., destinations) are on a network or link directly connected to the Remote Annex. The Remote Annex examines once and discards routes outside the **annex...end** blocks if the destination is not a directly connected network or link.

Setting Port Parameters for a PPP Interface

This section describes how to set port parameters for different PPP configurations (for more information on Remote Annex configuration parameters, see *Configuration Parameters* on page C-33).

Configurations for Connecting Two Subnets

Figure A-7 shows a LAN with two networks that are connected directly using subnets. Both ends of the PPP link have security enabled. Following the figure are the parameter settings required for this configuration.

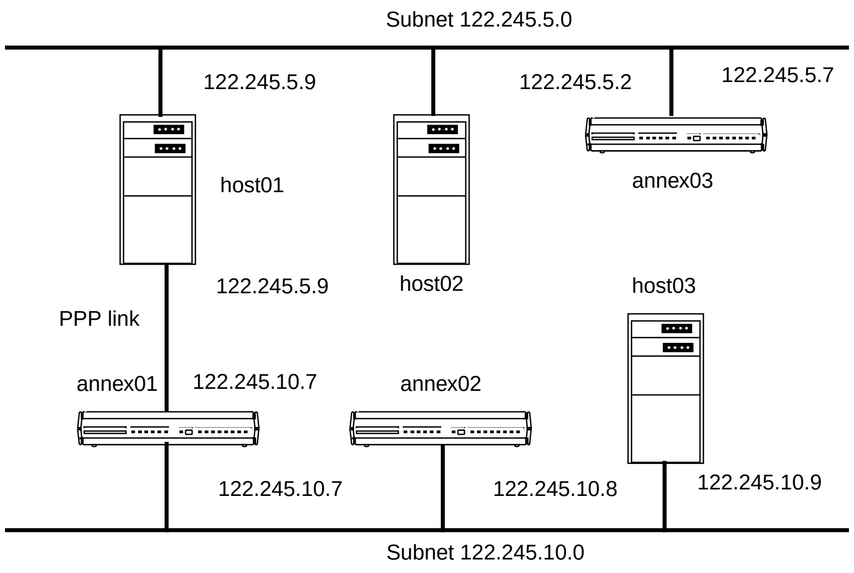


Figure A-7. PPP Link Connecting Two Subnets

- Set the **mode** parameter to **ppp**.
- Use the defaults for the **data_bits** (8), **stop_bits** (1), and **parity** (none) parameters.

  PPP is an 8-bit protocol. If **data_bits** is set to **7**, and **parity** is not set to **none**, the Remote Annex forces the **data_bits** setting to **8** and the **parity** setting to **none**. Otherwise, the Remote Annex *syslogs* an error message for the port.

- Set the **speed** parameter to the rate required for the PPP link. Set the remote end to match this value.
- To configure the port for hardware (EIA) flow control, set the **control_lines** parameter to **flow_control** and the **input_flow_control** and the **output_flow_control** parameters to **eia**.
- To configure the port for software (XON/XOFF) flow control, set the **control_lines** parameter to **none** and the **input_flow_control** and the **output_flow_control** parameters to **start/stop**. If you do this, you must also set the **ppp_acm** parameter 0x000a0000.
- Set the **address_origin** parameter to **local** (use the fixed endpoint address).
- Set the **remote_address** parameter to 122.245.5.9.
- Set the **local_address** parameter to 122.245.10.7.
- Set the **subnet_mask** parameter to 255.255.255.0.
- Set the **metric** parameter to one.
- Set the **ppp_username_remote** parameter to the string *castle*.
- Set the **ppp_password_remote** parameter to the string *sand*.
- Set the **ppp_security_protocol** parameter to **pap** (password authentication protocol).

- Set the **allow_compression** parameter to **Y** if you want the Remote Annex to accept compressed packets.
- Enter the routing information into the **gateway** section of the configuration file. For example:

```
%gateway
# PPP link to the 122.245.5.0 net
annex 122.245.10.7

     #122.245.5.9 is a gateway to the entire
     #122.245.5.0 net with a metric of 1
     route add 122.245.5.0 255.255.255.0 122.245.5.9 1

else

     #other Annexes will route to 122.245.5.0 via
     #122.245.10.7 with a metric of 2
     route add 122.245.5.0 255.255.255.0\
     122.245.10.7 2

end
```

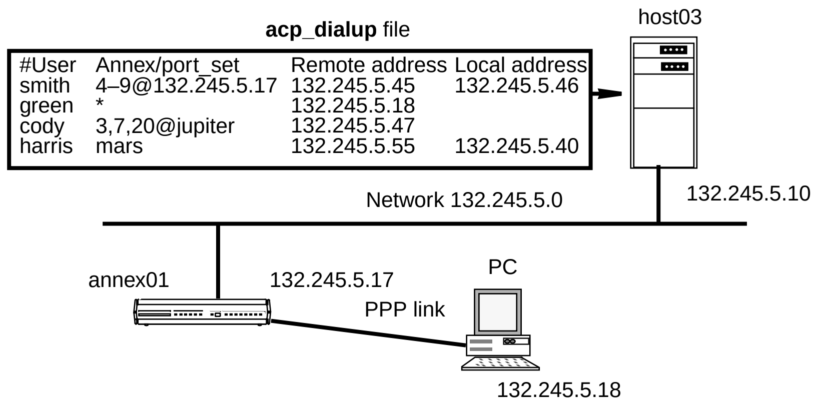
See *Creating gateway Entries in the Configuration File* on page A-350 for more details.

Configurations for Dial-in with Dial-up Addresses

Figure A-8 illustrates a configuration in which a single host connected to a Remote Annex through a PPP link appears to the network as an attached host. Following the figure are the parameter settings required for this configuration.

Setting up a port for dial-in PPP requires that you configure the port for both an inbound modem and a PPP link (see *Modems* on page A-99 for more details on inbound modems). Using a central security server for dial-in access allows users to have their own addresses.

Routes that cannot be reached until the dial-in PPP connection is activated are saved in the Remote Annex at boot time and activated when the PPP line is activated. When the PPP connection is terminated, these routes become inactive.



```
#User   Annex/port_set      Remote address Local address
smith   4–9@132.245.5.17    132.245.5.45   132.245.5.46
green   *                   132.245.5.18
cody    3,7,20@jupiter      132.245.5.47
harris  mars                132.245.5.55   132.245.5.40
```

Figure A-8. Connecting a Single Host Using PPP

- Set the **mode** parameter to **cli**, **auto_detect**, or **auto_adapt**.
- Setting the **type** parameter to **dial_in** registers the user with the **who** database as soon as the CLI process attaches to the line.
- Enable CLI and/or connection security using the port security parameters: **cli_security**, **connect_security**, and **port_password**.
- The **slip_ppp_security** parameter controls dial-in PPP access. If **enable_security** and **slip_ppp_security** are enabled, access to the PPP command is restricted via ACP and port access is logged in the ACP log file.

- Set the **ppp_security_protocol** parameter to **pap**, **pap-chap**, or **none**.
- Set the **ppp_username_remote** and **ppp_password_remote** parameters to the values expected by the remote node (the PC, in Figure A-8).
- Set the **allow_compression** parameter to **Y** if you want the Remote Annex to accept compressed packets.
- Use the supplied defaults for the **data_bits** (8), **stop_bits** (1), and **parity** (none) parameters.

  PPP is an 8-bit protocol. If **data_bits** is set to **7**, and **parity** is not set to **none**, the Remote Annex forces the **data_bits** setting to **8** and the **parity** setting to **none**. Otherwise, the Remote Annex *syslogs* an error message for the port.

- Set the **speed** parameter to the DTE speed of the modem on the serial port.
- To configure the port for hardware (EIA) flow control, set the **control_lines** parameter to **both** and the **input_flow_control** and the **output_flow_control** parameters to **eia**.
- To configure the port for software (XON/XOFF) flow control, set the **control_lines** parameter to **none** and the **input_flow_control** and **output_flow_control** parameters to **start/stop**. If you do this, you must set the **ppp_acm** parameter to 0x000a0000.
- Configure the **acp_dialup** file (see *Creating the acp_dialup File* on page 6-119).

- Set the **address_origin** parameter to **acp** so that the Remote Annex requests the endpoint addresses, based on the user's login, from ACP.
- You can leave **ppp_mru** parameters set to its default. If you are not using hardware (EIA) flow control, you can also leave the **ppp_acm** parameter set to its default.

You can also use the **acp_dialup** file to enable dynamic IP addressing by setting the remote address field of the file to **dhcp**. For complete information, see *Dynamic Allocation of Network Addresses* on page A-479.

After setting the parameters, dial into the port. If the port mode is **cli**, issue the **ppp** command at the CLI prompt. If the port mode is **auto_detect** or **auto_adapt**, press **Return** to enter **cli mode** and *then* issue the **ppp** command. For example, referring to Figure A-9, user *green* dials in from home to the Remote Annex. After passing CLI security, *green* issues the **ppp** command. The Remote Annex PPP asks the security server (132.245.5.10) for *green's* address. Then the Remote Annex negotiates with *green's* PC for this address and opens the link.

Configurations for Dial-in with Fixed Addresses

Figure A-9 illustrates a configuration in which a single host connected to a Remote Annex through a PPP link appears to the network as an attached host. Following the figure are the parameter settings required for this configuration.

Setting up a port for dial-in PPP requires that you configure the port for both an inbound modem and a PPP link (see *Modems* on page A-99 for more details on inbound modems). Using a central security server for dial-in access allows users to have their own addresses.

Routes that cannot be reached until the dial-in PPP connection is activated are saved in the Remote Annex at boot time and activated when the PPP line is activated. When the PPP connection is terminated, these routes become inactive.

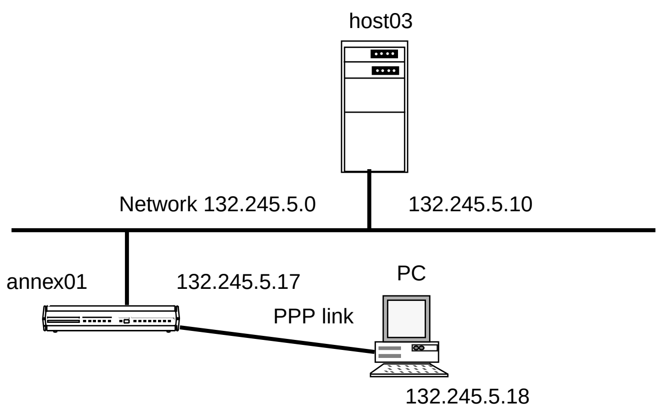


Figure A-9. Connecting a Single Host Using PPP with Fixed Addresses

- Set the **mode** parameter to **ppp**.
- Setting the **type** parameter to **dial_in** registers the user with the **who** database as soon as the CLI process attaches to the line.
- Enable CLI and/or connection security using the port security parameters: **cli_security**, **connect_security**, and **port_password**.
- The **slip_ppp_security** parameter controls dial-in PPP access. If **enable_security** and **slip_ppp_security** are enabled, access to the PPP command is restricted via ACP and port access is logged in the ACP log file.
- Set the **ppp_security_protocol** parameter to **pap**, **pap-chap**, or **none**.

- Set the **allow_compression** parameter to **Y** if you want the Remote Annex to accept compressed packets.
- Use the supplied defaults for the **data_bits** (8), **stop_bits** (1), and **parity** (none) parameters.

  PPP is an 8-bit protocol. If **data_bits** is set to **7**, and **parity** is not set to **none**, the Remote Annex forces the **data_bits** setting to **8** and the **parity** setting to **none**. Otherwise, the Remote Annex *syslogs* an error message for the port.

- Set the **speed** parameter to the DTE speed of the modem on the serial port.
- To configure the port for hardware (EIA) flow control and modem signals, set the **control_lines** parameter to **both** and the **input_flow_control** and the **output_flow_control** parameters to **eia**. If you do this, you must set the **ppp_acm** parameter to 0x000a0000.
- To configure the port for software (XON/XOFF) flow control, set the **control_lines** parameter to **none** and the **input_flow_control** and the **output_flow_control** parameters to **start/stop**.
- Set the **local_address** parameter to the Remote Annex's **en0** address.
- Set the **remote_address** parameter to the IP address of the remote node on the same network (132.245.5.18 in Figure A-11).
- Set the **address_origin** parameter to **local**.
- Leave the **ppp_acm** and **ppp_mru** parameters set to their defaults.

After setting the parameters, dial into the port. Your client should then begin LCP negotiation.

Configurations for Dial-out

Dial-out PPP allows system administrators to define a database of information about a modem pool and a set of virtual dial-out routes. The Remote Annex assigns each virtual dial-out route dynamically to ports in that pool. Each virtual dial-out route must be configured within the Remote Annex configuration file to start the PPP line (for more details, see *Dial-out Routes* on page A-392).

Dial-out routes are defined in the Remote Annex configuration file. *Setting Up the Configuration File* on page A-347 provides a sample configuration file; *Creating dialout Entries in the Configuration File* on page A-386 describes how to create **dialout** entries in the configuration file.

Dial-out PPP applies only to IP framing protocols; it does not apply to other protocols (e.g., ARAP, IPX, etc.).

Dial-out PPP also provides:

- On-demand dialing by logically tying the phone number to the virtual route.
- Dial-on-demand network routing by having the Remote Annex's dial-out routes configured with the proper static network routes (thus, the Remote Annex can route traffic to a particular network through a dial-out route's destination).
- The use of chat scripts (a sequence of commands that are used to log into the remote system after the phone connection is established and before IP forwarding begins; for more details, see *Chat Scripts* on page A-392).

Setting a port to provide dial-out PPP requires that you configure the port for both an outgoing modem and a PPP link.

Protocol Stack

Bringing up a PPP link includes three stages: link control protocol (LCP) negotiation, security, and NCP negotiation. The LCP establishes and negotiates the data link with the peer system. Next, an optional security phase authenticates the peer. Finally, NCP establishes and negotiates the network details and informs the Remote Annex that the interface is available.

Negotiating the LCP Options

The following subsections describe how the Remote Annex negotiates the LCP options.

Maximum Receive Unit (MRU)

The **ppp_mru** parameter sets the maximum receive unit (MRU). Acceptable values range from **64** to **1500**. The Remote Annex default is **1500**. The Remote Annex informs the peer that the **ppp_mru** parameter is its local MRU.

Asynchronous Control Character Map (ACCM)

The port parameter **ppp_acm** (asynchronous control mask) specifies which of the first 32 bytes (0x0 to 0x1F) can be sent as clear text and which should be protocol-escaped before being sent to the serial port.

The Remote Annex requests the **ppp_acm** parameter as its local mask. If the peer rejects **ppp_acm**, the Remote Annex accepts the hint if it is a superset of the Remote Annex's mask; otherwise, it uses the PPP default of 0xFFFFFFFF. The Remote Annex accepts any mask from the peer. Values range from **0x00000000** to **0xffffffff**. The Remote Annex default is **0x00000000** (unless either of the port parameters **input_flow_control** or **ouput_flow_control** are set to **start/stop**).

Setting the **ppp_acm** mask avoids sending characters that may bother the modems or devices through which the peers are connected. For example, if the modem uses CTRL-A (^A/0x01) as its attention character, it must be escaped before the Remote Annex sends it.

- **ppp_acm** for ASCII NUL (decimal 0) is 2 to the power of 0 = 0x00000001
- **ppp_acm** for ASCII SOH (decimal 1) is 2 to the power of 1 = 0x00000002
- **ppp_acm** for ASCII DC1 (decimal 17) is 2 to the power of 17 = 0x00020000
- **ppp_acm** for ASCII DC3 (decimal 19) is 2 to the power of 19 = 0x00080000

Thus, the mask for XON/XOFF (DC1 and DC3) equals the OR function of 0x00020000 and 0x00080000, or 0x000a0000.

When the Remote Annex sends an ACCM to the host, it follows this calculation to determine the initial value requested:

- The value set for **ppp_acm** (a 32-bit integer) is read in as the ACCM.
- If **input_flow_control** is set to **start/stop**, the following two additions are made:

  If **input_start_char** is 0–31 decimal, the bit indexed by this parameter is set in the ACCM.

  If **input_stop_char** is 0–31 decimal, the bit indexed by this parameter is set in the ACCM.
- If **output_flow_control** is set to **start/stop**, the following two additions are made:

  If **output_start_char** is 0–31 decimal, the bit indexed by this parameter is set in the ACCM.

  If **output_stop_char** is 0–31 decimal, the bit indexed by this parameter is set in the ACCM.

For example, the initial ACCM sent to the peer is 0x000A0001 if **ppp_acm** is set to 0x00000001 (i.e., the ASCII NUL character will not be sent) and the following parameters are set as indicated:

input_flow_control	**start/stop**
input_start_char	**^S**
input_stop_char	**^Q**
output_flow_control	**start/stop**
output_start_char	**f**
output_stop_char	**h**

Since the output flow control parameters are outside the range 0–31 decimal, they do not affect the ACCM.

The **na/admin** command **show port ppp_acm** still displays the **ppp_acm** setting. The CLI command **netstat –ip***nn*, where *nn* is the port number, displays the true mask (ACCM) value, i.e., the value negotiated between the two PPP processes.

Magic Numbers

The Magic Number option detects data-link anomalies, namely loopback. The Remote Annex always requests this option by sending a random 4-byte word out as its Magic Number in an LCP *Configure Request*.

Link Quality Monitoring (LQM)

The Remote Annex will not request LQM. It rejects any attempts by the remote peer for LQM and hints for the PPP default of none.

Protocol Field Compression (PFC)

PFC compresses the two-byte Asynchronous HDLC protocol field to one byte. The Remote Annex always requests and accepts PFC from the peer. If rejected, it accepts the PPP default of off. If the peer does not request PFC, the Remote Annex hints for PFC on. If the peer rejects this hint, the Remote Annex accepts PFC off.

Address and Control Field Compression (ACFC)

ACFC deletes non-ambiguous constant address and control fields in the Asynchronous HDLC headers. The Remote Annex always requests, and accepts, ACFC. If rejected, it accepts the PPP default of off. If the peer requests ACFC off, the Remote Annex hints for ACFC on. If the peer rejects this hint, the Remote Annex accepts ACFC off.

Negotiating the Network Control Protocol

The Remote Annex supports the following NCPs: AppleTalk Control Protocol (ATCP), Internet Packet Exchange Protocol Control Protocol (IPXCP), Internet Protocol Control Program (IPCP), CCP (Compression Control Protocol for PPP links), and Multilink PPP (MP). NCP options are negotiated in the same way as LCP options. An NCP peer opens the link and the interface is available to the Remote Annex. (For information on Multilink PPP, see the *Multilink PPP Addendum.)*

To specify one or more NCPs for a port, set the **ppp_ncp** port parameter to **ipxcp**, or any combination of **ipxcp**, **ipcp**, **atcp**, **mp**, and **ccp**. Separate multiple values with a commas. You can also specify **all** to indicate all of the protocols, which is the default.

Negotiating Data Compression

If you specify **ccp** as an NCP, the Remote Annex automatically requests data compression for a PPP link. Three types of compression are negotiated:

- Predictor-1, a public-domain algorithm
- BSD-Compress, a freely-available portion of the BSD UNIX sources
- STAC (with Check Modes 1, 3, and 4), a compression scheme used by the Windows ‘95 Dial-up Networking feature

Authentication Type

The authentication type specifies the style of authentication. The Remote Annex supports two authentication protocols for PPP:

- Password Authentication Protocol (PAP)
- Challenge-Handshake Protocol (CHAP)

Both of these protocols are run over the PPP link after the LCP negotiations are complete.

The Remote Annex can require the peer to pass a security check before starting NCP. The Remote Annex negotiates for the security specified by the **ppp_security_protocol** parameter. Valid arguments for this parameter are:

- **pap** (password authentication protocol [PAP]).
- **chap** (challenge-handshake protocol [CHAP]).
- **pap-chap** (first negotiate for CHAP; if peer NACKs, negotiate for PAP).
- **none** (do not negotiate; the default).

The Remote Annex negotiates an authentication request from a peer only if the **ppp_password_remote** and **ppp_username_remote** parameters are set for this port. If the peer refuses a negotiation request from the Remote Annex, the Remote Annex closes the link.

For a complete description of the Remote Annex's implementation of these protocols, see *Using PPP Security* on page A-514.

Negotiating the IP Address

The Remote Annex and the peer negotiate the IP address to be used on both sides of the link. Any address sent as zero requests that the peer set the address. Four parameters control the Remote Annex IP address negotiation: **address_origin**, **local_address**, **remote_address**, and **enable_security**.

If **address_origin** is set to **acp**, the Remote Annex makes an ACP **dialup_address()** call for the addresses to be used from the **acp_dialup** file.

If **address_origin** is set to **dhcp**, or if it is set to **acp** and the remote address field of the **acp_dialup** file is set to **dhcp**, the Remote Annex receives a dynamically assigned IP address.

If **address_origin** is set to **local** (its default value), or DHCP and ACP are not available, the Remote Annex defaults to using the **local_address** and **remote_address** as the addresses. The Remote Annex allows the other side of the link to select addresses only if these addresses are zero.

The Remote Annex uses two methods to negotiate the IP addresses. The preferred technique is to use the NCP type 3 *IP-Address* option. If the peer rejects this style of address negotiation, the Remote Annex falls back to using the deprecated NCP type 1 *IP-Addresses* option.

In either case, the Remote Annex requires the peer to use both the local and remote address of the Remote Annex. To allow the peer to select addresses, the Remote Annex addresses must be set to zero.

If each end has a zero address and the peer cannot provide both, or the Remote Annex has a non-negotiable address, the Remote Annex and the peer will never agree upon an address, and the link will fail to come up.

Negotiating the Compression Type

The Remote Annex and the peer negotiate a specific protocol compression TCP/IP Header. The options are VJ TCP/IP and none. If the **allow_compression** parameter is set to **Y**, the Remote Annex always negotiates for compression for both sides of the connection. If the peer is unable to comply, the Remote Annex defaults to the uncompressed option. If **allow_compression** is set to **N**, the Remote Annex never requests, and always rejects, TCP/IP header compression (the default is **N**).

BOOTP Requests

BOOTP is a bootstrap protocol that allows a diskless client to determine its Internet address, the Internet address of the server, and the name of the file to be loaded into memory. The Remote Annex ROMs use BOOTP to obtain boot information without requiring any manual set up on the Remote Annex.

If a diskless client sends a BOOTP request to the Remote Annex over a PPP line, the Remote Annex responds with its current local address, remote address, and boot host.

For more details on BOOTP, see the appropriate Remote Annex hardware installation guide.

Chapter 7
Serial Line Internet Protocol (SLIP)

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 connect a node to the network without requiring special interface hardware. The Remote Annex implementation of SLIP is compatible with the 4.3BSD implementation. The Remote Annex also supports Compressed SLIP as an option.

The Remote 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 nodes over serial lines.
- You can attach a PC to a Remote Annex serial port. Using SLIP as the network interface, the PC becomes a node 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 Remote 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 a Remote Annex. You can define a list of SLIP links and the local network interface over which a download is to occur. During a load, the Remote Annex makes a request from the first interface in the list. If the node associated with that interface does not respond, the Remote 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 Remote Annexes on the network. Any serial port can be configured for SLIP.

For best results, when using a SLIP link for booting or retrieving files from a host, set the line speed to at least 4800 bps.

You cannot boot a Remote Annex 2000 via SLIP using Port 1.

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 Remote 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 IP addresses to both ends of the SLIP link.

Connecting Two Networks Together

When connecting two networks together, you can:

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

The following figures illustrate Class B subnetted networks. Figure A-10 illustrates a network with a separate subnet address. In this example, the SLIP link is assigned a separate network address of 132.245.99.0 and a subnet mask of 255.2555.255.0; the link is then treated like any other physical network.

If you are conserving network numbers, you may prefer the configuration depicted in Figure A-11. Using this option, the IP addresses assigned to the end-points of the SLIP link are the hosts’ primary network IP addresses.

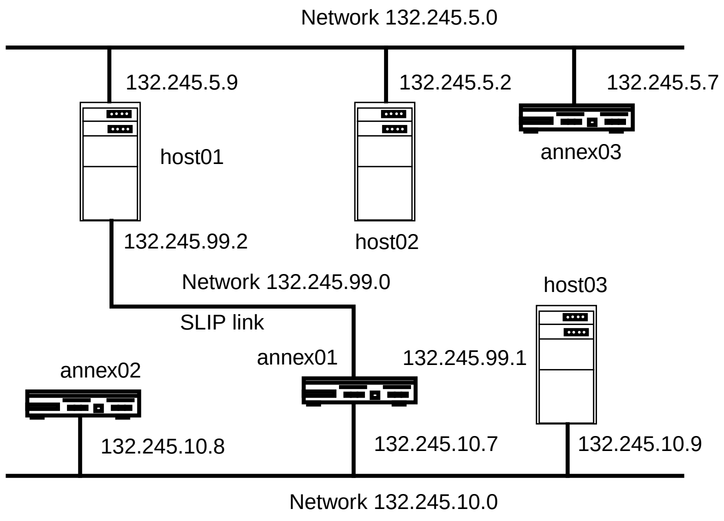


Figure A-10. SLIP Link with Separate Network Address

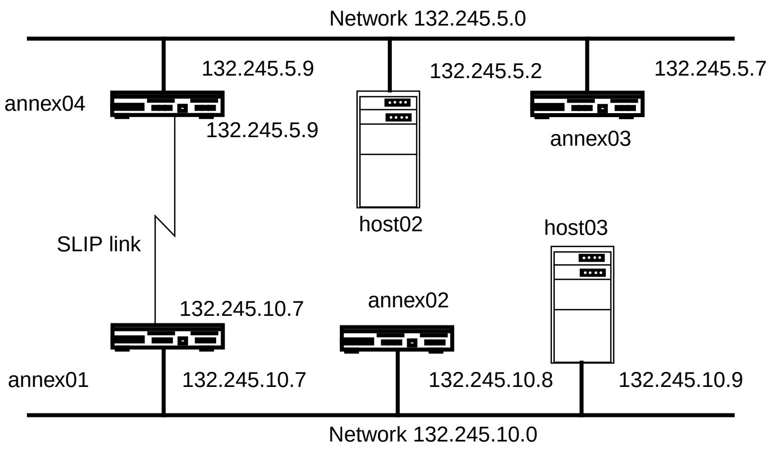


Figure A-11. SLIP Link with Two IP Addresses

Connecting a Single Host with a SLIP Link

In Figure A-12, a single PC connected to a Remote Annex through a SLIP link appears to the network as an attached host. Assign that PC a unique network host address.

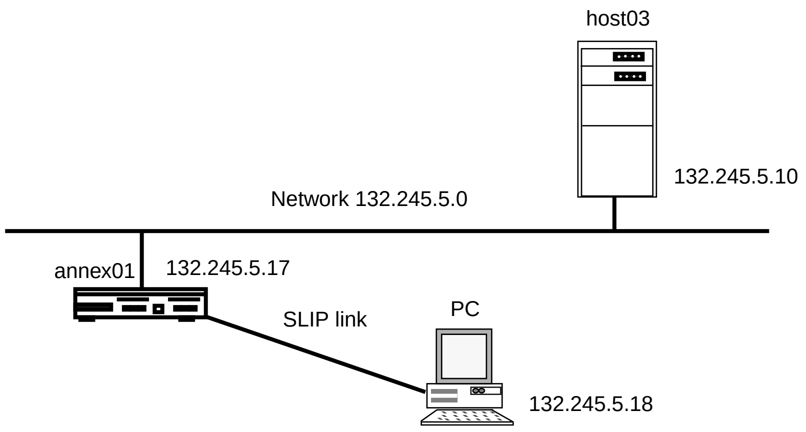


Figure A-12. Connecting a Single Host Using SLIP

Connecting a Remote Host

In Figure A-13, a remote host connected to a Remote Annex through a SLIP link appears as a local Remote Annex to the network. Assign the Remote Annex a unique network host address.

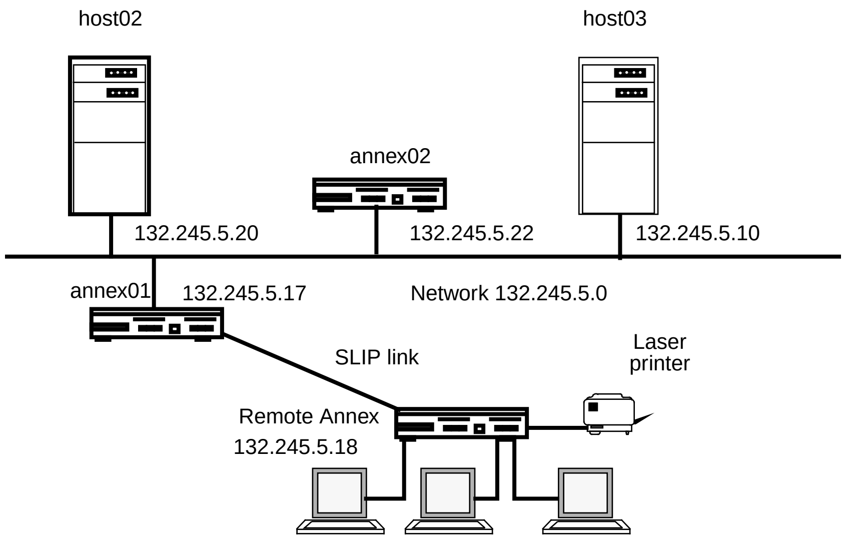


Figure A-13. Connecting a Remote Annex

Configuring Ports for a SLIP Interface

This section describes how to set port parameters for different SLIP configurations. After setting the parameters, issue the **slip** command at the CLI connection.

Setting SLIP Port Parameters

Set SLIP parameters using the **set port** command. The **show port slip** command displays these parameters.

- The **local_address** parameter defines the IP address for this port. This address is the address for the Remote Annex's side of the SLIP link. For example, in Figure A-11, the **local_address** for *annex01* is 132.245.99.1. In Figure A-12 the **local_address** for *annex01* is 132.245.10.7, which is the same address specified with the Remote Annex's **inet_addr** parameter.
- The **remote_address** parameter defines the IP 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. Although the address must be unique for the network, it can be assigned arbitrarily.
- The **subnet_mask** parameter defines the IP subnet mask if the SLIP link is using a subnet. For example, in Figure A-10 and Figure A-11, the **subnet_mask** setting is 255.255.255.0.
- The **metric** parameter defines the cost of getting to the remote end of the SLIP link (in relation to the cost of using other Remote Annex interfaces). This is referred to as the *hop count*. You may want to increase this number if the Remote Annex has a preferred (for example, faster) route to the remote host. If you do set the metric to a value greater than 1, remember that you may be decreasing the usable network diameter, since 15 is the highest valid **metric** value.
- The **allow_compression** parameter enables the Remote Annex to accept compressed packets.

- The **slip_no_icmp** parameter discards any ICMP packets that are destined for the SLIP link, thereby reducing unnecessary traffic and messages over the SLIP link.
- The **slip_mtu_size** parameter sets the maximum transmission unit (MTU) size on a compressed SLIP (CSLIP) port to either **large** (1006) or **small** (256).
- The **slip_tos** parameter causes the Remote 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.
- The **slip_ppp_security** parameter controls dial-up SLIP access. If **enable_security** and **slip_ppp_security** are enabled, access to the SLIP command is restricted via ACP and port access is logged in the ACP log file.
- The **address_origin** parameter enables the Remote Annex to request the endpoint address, based on the user's login, from ACP.

To use the SLIP link for booting and/or dumping, set the following parameters:

Using SLIP for Booting/Dumping

- The **slip_load_dump_host** parameter defines the IP address from which the Remote Annex requests boots and to which the Remote Annex dumps. This parameter can match the IP address specified in the **remote_address** parameter. You must set **slip_load_dump_host** when using a SLIP link for downloading software.

  Use **slip_load_dump_host** in place of the Remote Annex parameter **pref_load_host** if the port is defined with the Remote Annex parameter **load_dump_sequence**.

- Set the Remote Annex parameter **load_dump_sequence** 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 **asy***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, enter:

  ```
  command: set annex load_dump_sequence asy2,asy16,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 through which the Remote Annex is booting. Use **pref_load_host** with the local network interface; use **slip_load_dump_host** with 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 Remote 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 Remote Annex can take about ten minutes.

To download the operational code, but prevent dumps:

Downloading Operational Code without Dumping

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

Configuring SLIP for a Host

- Set the **mode** parameter to **slip**, **cli**, **adaptive**, **auto_detect**, or **auto_adapt** (see *Port Mode* on page A-53), with the following restrictions.
  - If you are dialing in with a SLIP line, you can only set the mode to **slip**.
  - If you set the mode to **cli mode**, **adaptive**, **auto_detect**, or **auto_adapt**, you may have to press **Return** at the time you are connected. This puts you in **cli mode**. You must then issue the CLI **slip** command to convert the port to **slip** mode. *The Remote Annex does not detect SLIP.*
- Setting the **type** parameter to **hardwired** registers the user with the **who** database according to the **input_is_activity** and **output_is_activity** parameter settings. If neither parameter is set, any user on this port is invisible to **who**.

  If the **input_is_activity** parameter is set, when the user enters data, the line is registered with the **who** database (generally used for hardwired CLI terminals).

  If the **output_is_activity** parameter is set, the line is registered when the Remote Annex first sends data (generally used for hardwired printers or other slave devices). This entry is removed on a hang-up or when the slave line is released, regardless of the **input_is_activity** and **output_is_activity** parameter settings.
- Set the **speed** parameter to the rate required for the SLIP link. Set the remote end to match this value. You cannot set the **speed** parameter to **autobaud** if the **mode** parameter is set to **slip**.

- To 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**.
- Use the supplied defaults for the **data_bits** (8), **stop_bits** (1), and **parity** (none) parameters.

SLIP is an 8-bit protocol. If **data_bits** is set to **7**, and **parity** is not set to **none**, the Remote Annex forces the **data_bits** setting to **8** and the **parity** setting to **none**. Otherwise, the Remote Annex generates an error message for the port.

Configuring SLIP for Dial-in

Setting a port to provide dial-in SLIP requires that you configure the port for both an incoming modem and a SLIP link. Setting the following parameters configures the designated ports to power-up as dedicated SLIP lines.

Routes that cannot be reached until the dial-in SLIP connection is activated are saved in the Remote Annex at boot time and activated when the SLIP line is activated. When the SLIP connection is terminated, these routes become inactive.

Configuring Ports for Incoming Modem Connections

- Set the **mode** parameter to **slip**, **cli**, **auto_detect**, **auto_adapt**, or **adaptive** (see *Port Mode* on page A-53). If you set the mode to **cli**, enter a carriage return <CR> to reach the CLI prompt. Then issue the **slip** command.
- Set the **speed** parameter to the same value as the speed of the modem serial port, e.g., **38400**.

- Set the **type** parameter to **dial_in** (this registers the user with the **who** database as soon as the CLI process attaches to the line).
- Set the endpoint addresses. One way to do this is to set the **local_address** and **remote_address** parameters**.** Another method is to set the **address_origin** parameter to **acp** so that the Remote Annex requests the endpoint addresses, based on the user's login, from ACP. (The **address_origin** parameter overrides the **local_address** and **remote_address** parameters.)
- Set up EIA/hardware flow control (RTS/CTS):

  Set the **input_flow_control** and **output_flow_control** parameters to **eia**.

  Set the **control_lines** parameter to **both**. The Remote Annex asserts RTS when it is ready to receive data, checks the CTS input before transmitting data, and causes the port to be reset when the modem hangs up.
- The **do_compression** parameter forces TCP header compression on the SLIP link. When enabled, the Remote Annex always sends compressed packets.
- The **cli_security** parameter enables CLI security for a dial-in SLIP line.
- If both the **enable_security** and **slip_ppp_security** parameters are enabled, access to the **slip** command is restricted via ACP and port access is logged in the ACP log file.
- Use the supplied defaults for the **data_bits** (8), **stop_bits** (1), and **parity** (none) parameters.

  SLIP is an 8-bit protocol. If **data_bits** is set to **7**, and **parity** is not set to **none**, the Remote Annex forces the **data_bits** setting to **8** and the **parity** setting to **none**. Otherwise, the Remote Annex generates an error message for the port.

After setting the port parameters, reset the appropriate port(s) or Remote Annex subsystem, or reboot the Remote Annex for the changes to take effect.

A sample command line for configuring ports 1 through 4 with the minimum required parameters for this set up looks like this:

```
admin: set port=1-4
admin: set port mode slip
admin: set port speed 38400
admin: set port control_lines both
admin: set port input_flow_control eia
admin: set port output_flow_control eia
admin: set port type dial_in
```

Configuring Ports for SLIP mode

If you want to use these SLIP lines as normal CLI dial-up lines, i.e., entering the **slip** command puts the lines into SLIP mode, set the **mode** parameter to **cli**, **adaptive**, **auto_detect**, or **auto_adapt**:

```
admin: set port=1-4
admin: set port mode cli
admin: set port speed 38400
admin: set port control_lines both
admin: set port input_flow_control eia
admin: set port output_flow_control eia
admin: set port type dial_in
```

Configuring SLIP for Dial-out

Dial-out SLIP allows system administrators to define a database of information about a modem pool and a set of virtual dial-out routes. The Remote Annex assigns each virtual dial-out route dynamically to ports in that pool. Each virtual dial-out route must be configured within the Remote Annex configuration file to start the SLIP line (for more details, see *Dial-out Routes* on page A-392).

Dial-out routes are defined in the Remote Annex configuration file. *Setting Up the Configuration File* on page A-347 provides a sample configuration file; *Creating dialout Entries in the Configuration File* on page A-386 describes how to create **dialout** entries in the configuration file.

Dial-out SLIP applies only to IP framing protocols; it does not apply to other protocols (e.g., ARAP, IPX, etc.).

Setting a port to provide dial-out SLIP requires that you configure the port for both an outgoing modem and a SLIP link (see *Configuring SLIP for Dial-in* on page 7-147 for sample parameter settings).

Set the **mode** parameter to **slave**, **adaptive**, or **auto_adapt**.

Dial-out SLIP also provides:

- On-demand dialing by logically tying the phone number to the virtual route.
- Dial-on-demand network routing by having the Remote Annex's dial-out routes configured with the proper static network routes (thus, the Remote Annex can route traffic to a particular network through a dial-out route's destination).
- The use of chat scripts (a sequence of commands that are used to log into the remote system after the phone connection is established and before the IP framing protocol begins; for more details, see *Chat Scripts* on page A-392).

Routing Across a SLIP Link (Basic Passive RIP)

Both active and passive routing are available via the Routing Information Protocol (RIP) on the Remote Annex. The following sections deal with using only the most basic features of passive RIP and are intended for administrators who need minimal routing features. For complete information on both passive and active RIP, see *IP Routing* on page A-169.

Both active and passive RIP are enabled by default. To turn off active RIP, set the interface parameter **rip_advertise** to **none** for all Remote Annex interfaces. See *rip_advertise* on page A-222.

The Remote Annex bases its routing table on the information you specify in the **gateway** section of the configuration file. As a passive gateway, the Remote Annex then updates the table according to information it receives from other routers but does not broadcast routing information itself. This means that a Remote Annex with a SLIP interface forwards packets addressed to the host at the remote end of the connection, but does not inform other hosts, routers, or Remote Annexes that it has this capability. Other hosts and routers on the same network must be told about the route before they can use it.

To guarantee that a route in the **gateway** section of the **config.annex** file (or in the gateway entry in **/etc/gateways**) uses a particular SLIP interface, the next hop in the route must match the remote address of the SLIP link.

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 Remote Annex. Create an entry in a host's **/etc/gateways** file. Using the example in Figure A-11 on page A-140, *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 Remote 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 Remote 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 Remote Annexes aware of a route using a SLIP link, create a **gateway** entry in the Remote Annex configuration file (see *Creating gateway Entries in the Configuration File* on page A-350 for more details). Using Figure A-10 on page A-140, the entries for the Remote Annexes on network 132.245.10.0 are:

```
annex 132.245.10.7

     route add 132.245.5.0 255.255.255.0 132.245.99.2 1 else

     route add 132.245.5.0 255.255.255.0 132.245.10.7 2
     route add 132.245.99.2 255.255.255.0 132.245.10.7 1 end
```

These entries inform *annex01* that *host01* is a gateway to network 132.245.5.0 (with a metric of 1) and inform other Remote Annexes on network 132.245.10.0 that *annex01* is a gateway to either *host01*(with a metric of 1) or network 132.245.5.0 (with a metric of 2).

Route Cache

The route cache is a list of routing entries stored by the Remote Annex. When the Remote Annex boots, the route cache is created from the **annex...end** and **subnet...end** blocks in the gateway section of the configuration file (see *Creating gateway Entries in the Configuration File* on page A-350 for more details). When **routed** starts, entries in the route cache are added to the routing table if their next hops are on a network directly connected to the Remote Annex.

Extending a Single Host onto the Network

The Remote Annex can use Proxy-ARP to attach a single host and Remote Annexes onto the network transparently. Using Proxy-ARP, the Remote Annex answers 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 appear on the Remote Annex for the SLIP interface in Figure A-12:

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

Typically, a Proxy-ARP is used when the Remote Annex's SLIP link is to a single device, i.e., both the device and the Remote Annex use the same Internet network address. No other routing information is required with this configuration (see *arp* on page C-135 for more information on manipulating the ARP cache).

The destination address of the SLIP link must be on the same network as the Remote Annex.

BOOTP Requests

BOOTP is a bootstrap protocol that allows a diskless client to determine its Internet address, the Internet address of the server, and the name of the file to be loaded into memory.

- The Remote Annex ROMs use BOOTP to obtain boot information without requiring any manual set up on the Remote Annex.
- If a diskless client sends a BOOTP request to the Remote Annex over a SLIP line, the Remote Annex responds with its current local address, remote address, and boot host (the *Annex Hardware Installation Guide* describes BOOTP in detail).

Chapter 8
Dial-up Networking

In dial-up networking, users gain remote access to a local area network using modems and dial-up telephone lines. The Remote Annex implementation of protocols such as SLIP and PPP provides dial-in connectivity in a multi-protocol network. Using the Remote Annex as a dial-up server, a remote user can dial into a modem connected to the Remote Annex and become a directly connected network node; the Remote Annex then is transparent to the user. The Remote Annex can also generate a call to the remote server and become a directly connected network node while remaining transparent to the user.

For details on using modems, see *Modems* on page A-99.

For details on using a SLIP link, see *Serial Line Internet Protocol (SLIP)* on page A-137.

For details on using a PPP link, see *Point-to-point Protocol (PPP)* on page A-111.

For details on filtering, see *Filtering* on page A-249.

Dynamic Dialing

Dynamic dialing allows system administrators to define a database of information about a modem pool and a set of virtual dial-out routes. The Remote Annex assigns each virtual dial-out route dynamically to ports in that pool. Each virtual dial-out route must be configured within the Remote Annex configuration file to start the dial-out connection (for more details, see *Dial-out Routes* on page A-392).

Dial-out routes are defined in the Remote Annex configuration file. *Setting Up the Configuration File* on page A-347 provides a sample configuration file; *Creating dialout Entries in the Configuration File* on page A-386 describes how to create **dialout** entries in the configuration file.

Dynamic dialing applies only to IP framing protocols.

Dynamic dialing also provides:

- Dial-on-demand network routing by having the Remote Annex's dial-out routes configured with the proper static network routes (thus, the Remote Annex can route traffic to a particular network through a dial-out route's destination).
- The use of chat scripts (a sequence of commands that are used to log into the remote system after the phone connection is established and before IP forwarding begins; for more details, see *Chat Scripts* on page A-392).
- The use of both the **netact** and **no_start** filter actions for dial-out connection interfaces. These actions will operate as described in *Filtering* on page A-249. However, you cannot add a filter by using the CLI **filter** subcommand **add**. Instead, you specify a *filter* definition in the **dialout** section of the Remote Annex configuration file. In this definition, omit both the word **add** and the name of the interface (e.g., asy7). For more details, see *Setting Up the Configuration File* on page A-347.

A dynamic dialing route appears as a normal route to the end user. It has an entry in the route cache and is advertised by the Remote Annex. (Advertising is enabled by default; see *rip_advertise* on page C-98). When a user tries to send traffic to its destination (e.g., using **telnet**), the connection protocol's process detects this traffic, establishes the phone connection by dialing into a modem, and then continues normal operation.

Network Inactivity

Dynamic dialing resets the dial-out line when no traffic occurs on the line for a certain length of time. Resetting the line terminates the phone connection, saving costs by stopping inactive users from keeping phone connections open.

With active RIP enabled (as it is by default) on a dialout route, RIP updates, sent every 30 seconds, are considered activity, and reset the inactivity timer. A filter must be applied to the **dialout** configuration to prevent this. A sample filter entry for a **dialout** configuration is:

```
filter in exclu proto udp src_port router netact
filter out exclu proto udp src_port router netact
```

Additionally, RIP updates are not sent over an inactive/quiescent dialout line, and therefore do not activate an inactive/quiescent dialout line.

Enabling Dynamic Dialing

To enable dynamic dialing:

1. **Obtain a valid** dialout **value for the Remote Annex** option_key **parameter. Each Remote Annex requires a unique value. The way to obtain a key depends on the configuration and type of your Remote Annex. Some** option_ key **values are physically attached to the underside of the Remote Annex. If this is true for your Remote Annex, enter this value as described in Step 2.**

   If the **option_key** value is not attached to your Remote Annex, contact your supplier to obtain a key. You will need to specify the Ethernet address of your Remote Annex; it is taped to the back of the unit.

   The **option_key** parameter enables a variety of Remote Annex features, including tn3270, AppleTalk, and IPX, depending upon what you specified when you ordered your Remote Annex.

The **dialout option_key** also enables filtering.

To determine which options are enabled, issue the CLI **stats –o** command:

```
annex: stats –o
```

```
KEYED OPTIONS:

                         LAT: keyed off
                       Atalk: keyed off
                      tn3270: keyed off
          dialout/filtering: keyed off
                         IPX: keyed off
MODULES DISABLED
     None
```

The *MODULES DISABLED* field indicates the current setting of the **disabled_modules** parameter. If *dialout* is listed as disabled, you cannot use it, even if the **option_key** for it is set correctly (see *disabled_modules* on page C-57.)

Although LAT is an option, you set it using the Remote Annex **lat_key** parameter (for more details, see *lat_key* on page C-67).

2. **Use** na, admin**, or SNMP to set the** option_key **parameter. For example, if your key value is *RaqbDwv8e*, enter the following:**

```
annex: su
Password:
annex# admin
Annex administration Remote Annex R13.3, 72 ports
admin: set annex option_key RaqbDwv8e
```

3. **Reboot the Remote Annex to put the** option_key **setting into effect:**

```
admin: q
annex# boot
```

4. **Reconnect to the Remote Annex and issue the CLI** stats -o **command to make sure** dialout **is keyed on.**

5. **Configure the** modem **section of the Remote Annex configuration file. Table A-41 on page A-375 lists the field definitions for** modem **entries (for more details, see *Creating modem Entries in the Configuration File* on page A-374).**

Several standard entries for the **modem** section of the Remote Annex configuration file are supplied with the software distribution. These entries, defined for use with the configuration file, are located in the file **/usr/annex/bfs/modems.annex**.

If the modem you are using is contained within this file, using the **%include** *filename* command tells the parser that entries in the specified file are part of the configuration file. For example:

```
%modem
%include modems.annex
```

If the modem you are using is not contained within this file, you must edit the configuration file accordingly.

On each Remote Annex, edit the configuration file to contain a **modem** section similar to the following sample entry. If the Remote Annex boots from a host, the file resides on the host; for self-boot units, the file resides on the Remote Annex.

A typical modem entry looks like this:

```
%modem
# Acme Widget and Modem Corp Type 1234 V.34 modem

type_of_modem   ACME_288
ready_status    0
connect_status  1 5 10 13 18 20 21 25 43 47\
                85 91 99 103 107
connect_abort   4 6 12
connect_retry   3 7 8
connect_ignore  11
reset_cmd       AT&F1
reset_delay     3.0+0.0
dialout_setup_cmd
ATE1Q2V0X6&A0&B1&C1&D2&H1&I0&K1&R2S0=0
dialin_setup_cmd
ATE1Q2V0X6&A0&B1&C1&D2&H1&I0&K1&R2S0=1
dial_cmd        ATDT
timeout         60
retry           3
end
```

In the above modem configuration, the *dialin_setup_cmd* and *dialout_setup_cmd* field entries appear on two lines. This is a limitation of the printed page. Enter the actual command strings as one continuous entry without any line breaks.

6. Reset the modem:

```
annex# admin
Annex administration Remote Annex 13.3, 72 ports
admin: reset annex modem
```

7. **Configure the** dialout **section of the Remote Annex configuration file. Table A-44 on page A-387 provides field definitions for the dialout entries.**

   Each entry in the **dialout** section of the configuration file defines a dial-out route. The format of a **dialout** entry looks like this:

```
%dialout

global_timeout <time-out value>

# this is a comment line

begin_route      <route id>
local            <local address>
remote           <remote address>
mode             <slip or ppp>
ports            <port set/rotary>
phone            <phone number>
chat             <chat script list>
filter           <filter command>
disabled         <time interval>
advertise        <Y or N>
set              <parameter_name setting>
set              <parameter_name setting>
end_route
```

   The mandatory fields for a **dialout** entry are: *begin_route*, *remote*, *mode* (**slip** or **ppp**), *ports*, *phone*, and *end_route*.

   If you specify a rotary for ports, the telephone number defined for the rotary takes precedence over the phone number specified in the **dialout** entry.

   Using separate *set* field entries, set any port parameters that you want included in the **dialout** entry (for more details, see Table A-45 on page A-390).

   Any port parameter not set in the **dialout** entry will use the current settings in non-volatile memory.

Set the following parameters on each port to which the **dialout** entry applies:

a) Set the **mode** parameter to **slave**, **adaptive**, or **auto-adapt**.

b) Set **type** as appropriate for the attached equipment.

c) Set **speed** as appropriate for the attached equipment.

d) Set the **type_of_modem** parameter to match the *type_of_modem* field entry in the **modem** section of the Remote Annex configuration file.

e) Set **control_lines** as appropriate for the attached equipment.

f) Set **input_flow_control** as appropriate for the attached equipment.

g) Set **output_flow_control** as appropriate for the attached equipment.

For more details, see *Creating dialout Entries in the Configuration File* on page A-386. For more details on Remote Annex configuration parameters, see *Configuration Parameters* on page C-33.

8. **Issue a** reset annex dialout **command.**

If a SLIP or PPP line is using both dynamic dialing and network inactivity, network inactivity only goes into effect after dynamic dialing has been activated.

9. **Initiate a dynamic dialing session.**

a) On the terminal connected to each Remote Annex, issue the CLI superuser **modem –a** command to verify the modem type and each of the strings defined for the modem.

b) On each Remote Annex, issue a CLI **netstat –r** command to verify that there is a route (e.g., do2).

c) When outbound traffic is detected, the Remote Annex initiates a dial-out connection based on the information in the **dialout** section of the Remote Annex configuration file. Debug level syslogging tracks the connection attempt.

d) If the network inactivity timer expires before the dialout connection is established, the Remote Annex breaks the connection. Any subsequent activity across the link (that has not been filtered out) re-establishes the dynamic dial connection.

See *Sample Configurations for Dynamic Dialing* on page 8-164 for an example of a final configuration for two Remote Annexes.

For details on filtering, see *Filtering* on page A-249.

For more details on dial-up SLIP, see *Serial Line Internet Protocol (SLIP)* on page A-137.

For more details on dial-up PPP, see *Point-to-point Protocol (PPP)* on page A-111.

For more details on editing the configuration file, see *Setting Up the Configuration File* on page A-347.

For more details on Remote Annex configuration parameters, see *Configuration Parameters* on page C-33.

Sample Configurations for Dynamic Dialing

This section illustrates a final configuration for two Remote Annexes configured for dynamic dial-out routing.

Router A’s **dialout** configuration:

```
%dialout
annex 132.245.1.1

begin_route   1
mode          ppp
local         132.245.1.1
remote        132.245.2.1
set           net_inactivity 20
phone         16175551234
set           do_compression Y
set           allow_compression Y
set           net_inactivity_units minutes
set           subnet_mask 255.255.255.0
set           ppp_ncp ipcp
set           rip_sub_advertise Y
set           rip_sub_accept Y
set           rip_advertise all
set           rip_accept all
advertise     y
ports         1
filter        in exclu proto udp src_port router netact
filter        out exclu proto udp src_port router netact
end_route
end
```

Router A's Port 1 (asy1) configuration (all other parameters at default values):

```
port asy1:

     Port Generic Parameters

mode: auto_adapt             location: "Corporate"
type: dial_in                term_var: ""
prompt: ""                   cli_interface: uci
speed: 115200                autobaud: N
data_bits: 8                 stop_bits: 1
parity: none                 max_session_count: 3
allow_broadcast: Y           broadcast_direction: port
imask_7bits: N               cli_imask7: Y
ps_history_buffer: 0         banner: Y
tcp_keepalive: 0             dedicated_address: 0.0.0.0
dedicated_port: telnet       type_of_modem: "pract_pm28800"
default_session_mode:interactive dedicated_arguments: ""

     Flow Control and Signal Parameters

control_lines: both          input_flow_control: eia
input_start_char: ^Q         input_stop_char: ^S
output_flow_control: eia     output_start_char: ^Q
output_stop_char: ^S         need_dsr: N
ixany_flow_control: N        backward_key: ""
forward_key: ""
```

Router B's **dialout** configuration:

```
%dialout
annex 132.245.2.1
begin_route     1
mode            ppp
local           132.245.2.1
remote          132.245.1.1
set             net_inactivity 20
phone           16175554321
set             do_compression Y
set             allow_compression Y
set             net_inactivity_units minutes
set             subnet_mask 255.255.255.0
set             ppp_ncp ipcp
set             rip_sub_advertise Y
set             rip_sub_accept Y
set             rip_advertise  all
set             rip_accept all
advertise       Y
ports           1
filter          in exclu proto udp src_port router netact
filter          out exclu proto udp src_port router netact
end_route
end
```

Router B's Port 1 (asy1) configuration (all other parameters at default values):

```
port asy1:

     Port Generic Parameters

mode: auto_adapt             location: "Field Office"
type: dial_in                term_var: ""
prompt: ""                   cli_interface: uci
speed: 115200                autobaud: N
data_bits: 8                 stop_bits: 1
parity: none                 max_session_count: 3
allow_broadcast: Y           broadcast_direction: port
imask_7bits: N               cli_imask7: Y
ps_history_buffer: 0         banner: Y
tcp_keepalive: 0             dedicated_address: 0.0.0.0
dedicated_port: telnet       type_of_modem: "zoom_28.8"
default_session_mode: interactive dedicated_arguments: ""
```

(continued on next page)

```
     Flow Control and Signal Parameters

control_lines: both            input_flow_control: eia
input_start_char: ^Q           input_stop_char: ^S
output_flow_control: eia       output_start_char: ^Q
output_stop_char: ^S           need_dsr: N
ixany_flow_control: N          backward_key: ""
forward_key: ""
```

Displaying Dynamic Dialing Routes in the Routing Table

The CLI **netstat –r** command displays statistics and information about all available routes in the routing table; Dynamic dialing routes that do not have a phone connection established appear without a **U** in the *Flags* field. A route comprises a destination host or network and the gateway through which data is forwarded.

In the following example, routes *dol4* and *dol6* are dynamic dialing routes waiting for activation:

```
annex: netstat -r
Routing tables
Destination     NextHop        Flags Usage  UseCount Mtr Interface
127.0.0.0/8     *              UI    fixed  0        2   lo0
132.245.1.0/24  132.245.44.22  UR    -114   0        3   en0
132.245.33.0/24 *              QI    fixed  147      1   do14
132.245.34.0/24 *              QI    fixed  0        2   do16
132.245.44.0/24 *              UHF   fixed  838      2   asy10(slip)
```

In the following example, dynamic dialing route *do14* has been dialed and used:

```
annex: netstat –r
Routing tables
Destination     NextHop        Flags Usage  UseCount Mtr Interface
127.0.0.0/8     *              UI    fixed  0        2   lo0
132.245.1.0/24  132.245.44.22  UR    -114   0        3   en0
132.245.34.0/24 *              QI    fixed  0        2   do16
132.245.44.0/24 *              UHF   fixed  838      2   asy10(slip)
132.245.33.0/24 *              UI    fixed  1        1   asy14(ppp)
```

For more information on the **netstat** command, see *Displaying Network Statistics* on page B-1 and *netstat* on page C-158.

Chapter 9
IP Routing

This chapter describes the Remote Annex implementation of Internet Protocol (IP) routing and the Routing Information Protocol (RIP) for SLIP, PPP, and Ethernet ports. The following topics are covered in this chapter.

- Prerequisites.
- Understanding Remote Annex IP routing and RIP. This includes sections on:
  - RIP versions.
  - Routing tables.
  - The difference between passive and active RIP.
  - Routing interfaces.
  - IP addressing.
  - Proxy ARP.

  Knowing how the Remote Annex implements these features is crucial to using the Remote Annex effectively as a full router.
- Overview of routing configuration parameters.
- Enabling, disabling, and configuring passive RIP without active RIP.
- Prerequisites for using active RIP.
- Configuring active RIP.
- Reference descriptions of all RIP configuration parameters.
- Displaying routing information.
- Troubleshooting.
- Other documentation.

Prerequisites

This chapter assumes that:

- Your Remote Annex is attached to the network and both are operational.
- Any hosts you wish to reach are attached to the Ethernet or the Remote Annex and links to them have been proven to work.
- Any modems you intend to use have been tested in the configuration in which you intend to use them. For example, testing a modem for dial-in and thereby assuming dial-out works is not sufficient.
- RIP is the primary routing application on your network and you know which routers are running it; you also know which RIP version (1 or 2) the routers are running.
- You understand RIP. Read the remainder of this chapter, and if you need more information, consult the sources cited at the end of the chapter.

Understanding IP Routing and RIP

IP routing is the process of determining the best path to follow to deliver a piece of data, contained in an IP *datagram*, to its destination on a TCP/IP network. Only a simple network, in which all systems directly attach to a single LAN, does not require routing. But a simple network can easily grow into an internet – a collection of multiple, interconnected networks – where routing is required to reach hosts on distant networks. Special machines called routers connect two or more networks to each other and route packets from one to the other.

Routers use routing applications to learn and generate routing information. The Remote Annex uses RIP, an application that runs on top of the transport-layer protocol UDP (User Datagram Protocol), for this purpose. Through information broadcast by other RIP routers, the Remote Annex maintains a routing table containing up-to-date routes to various destinations on local and remote networks. (This table also contains AppleTalk routes.)

Definition of a Route

The information a RIP router broadcasts contains the network addresses in the router's routing table and the number of routers (including itself) that must be crossed to reach these addresses. Each router is a *hop* in the path to the address; the next router in the path is called the *next hop*; and the total number of routers that must be crossed is called the *hop count* (also known as the *metric*). The next hop in a routing table is always on a network directly attached to the router containing the table. The RIP metric is an integer from 1 through 15, with 1 representing the best (shortest) route. Together, the IP address of a destination host or network, the next hop, and the metric constitute a *route*.

If one of the routers or networks in a particular RIP route fails, a time-out mechanism prevents other routers from keeping the route in their routing tables and advertising it as viable. If Router A's table contains a route learned from Router B, and three minutes elapse between Router A hearing from Router B, Router A marks the route in question as invalid by setting its metric to 16; this value is referred to as *infinity*. Router A advertises the route, with a metric of 16, for two more minutes, so that other routers can learn the route is unusable. At the end of that time, Router A deletes the route from its routing table.

Routing versus Forwarding

Do not confuse IP routing with IP forwarding. IP forwarding is the process of sending an IP datagram to the next hop on the way to its destination. IP routing is the algorithm that determines the next hop to use. The Remote Annex kernel performs IP forwarding; Remote Annex RIP performs routing.

Choosing Passive or Active RIP

The Remote Annex can run passive RIP alone, or both passive and active RIP. The default is both, but you can disable active RIP by changing the value of the **rip_advertise** parameter (see *Enabling Passive RIP Alone* on page 9-193). When running only passive RIP, the Remote Annex revises its routing table based on the routing updates it receives, but does not broadcast updates itself. When configured for active RIP (which is the default), the Remote Annex acts as a full router: it not only listens for updates but also broadcasts them every 30 seconds.

Using only passive RIP is appropriate when:

- Your network is small and does not connect to other networks. In this case, you can configure any necessary routing information by defining static routes. Static routes are routes you enter yourself, as opposed to those RIP learns over the network.
- Your network is large but has only one gateway to the other networks you need to access. In this case, you can save the overhead of broadcasting updates over all nodes. Instead, you can use passive RIP, configure static routes where needed, and define the gateway to other networks as the default route. The Remote Annex will use the default route if it knows no other route to a given destination.

In most other situations, active RIP is more useful than passive RIP.

The **routed** parameter controls whether any kind of RIP is enabled. Initially, **routed** is set to **Y** and both active and passive RIP are enabled. See *Enabling Passive RIP Alone* on page 9-193 for information on setting **routed** and disabling active RIP; see *Active RIP Prerequisites* on page 9-213 for information about active RIP requirements.

RIP Versions

The Remote Annex supports both RIP version 1 (STD 34, RFC 1058) and RIP version 2 (RFC 1388). RIP 2 is a backward-compatible extension of RIP 1 that expands the amount of information carried in RIP updates and authenticates them using passwords. RIP 2 broadcasts or multicasts updates, depending on whether or not updates are to be sent to RIP 1 as well as RIP 2 systems.

Not all routers support RIP version 2. Check the RIP versions running on the routers on your network before configuring RIP on the Remote Annex.

Route Cache and Routing Table

The Remote Annex stores IP routes in two places: the route cache and the routing table.

Route Cache

The route cache contains user-configured routing information – static routes, and sometimes a default route. The Remote Annex can learn default routes through RIP updates as well as from you, which is why all default routes are not necessarily cached.

The Remote Annex copies static and default routes to the route cache from the **gateway** section of the Remote Annex configuration file (see *Entering Routes in the Remote Annex Configuration File* on page 9-194) and/or from routes you define via the CLI superuser **route** command (see *Entering Routes using the route Command* on page 9-206).

Routes in the cache include those whose next hops are directly reachable – that is, up and running on a network directly connected to the Remote Annex – and those that are not yet reachable. Routes with reachable next hops are immediately copied to the RIP routing table. Routes whose next hops are not yet directly reachable are copied to the routing table as soon as their next hops become reachable. The latter technique saves the Remote Annex the trouble of consulting the configuration file, which typically is not stored on the Remote Annex, each time a route's status changes. A copy of the route remains in the routing cache, in case its next hop becomes unreachable again.

The route cache also contains back-up routes if two routes in the routing table have the same destination interface. In this case, one of the routes is removed from the routing table and stored in the cache, as follows:

- If one of the routes is for dial-out and the other is not, the dial-out route is moved to the cache.
- If the routes are both dial-out or both non-dial-out, the route with the highest metric is moved to the cache.
- If the routes are both dial-out or both non-dial-out and have the same metric, the route with the preferred interface type is stored in the routing table and the other in the cache. For example, en0 routes are preferred over serial interface routes.

To display the route cache, use the **netstat –C** command described in *Displaying the route cache* on page 9-233.

Routing Table

The RIP portion of the routing table contains routes for all IP destinations the Remote Annex can reach currently. This includes the local loopback route and the user-configured default route (if one exists), as well as routes copied from the route cache, routes learned by RIP, and routes to directly connected networks. The latter are called *interface* routes and are discussed in *Interface Routes* on page 9-178.

If a route becomes invalid (because its hop count reaches 16 or the interface to its next hop goes down), it is kept for two minutes in the routing table, so that the route's invalidity is advertised to other routers (when active RIP is enabled).

RIP uses the routing table for the updates it advertises (when active RIP is enabled).

To display the routing table, use the **netstat –r** command described in *Displaying Routing Information* on page 9-227.

How Hosts Learn Routing Information

Unlike routers, hosts do not run routing applications and do not maintain extensive routing tables. Host table entries tend to be restricted to information learned via ICMP messages generated by routers on the network. When the Remote Annex is acting as a router, it generates messages to provide hosts with the following information:

- Whether or not the Remote Annex is available as a router on the host's local network.
- Whether or not the Remote Annex knows of a better first hop than itself to a given destination.

- The size of the biggest packet the Remote Annex can forward without fragmentation. Fragmentation involves breaking a packet into pieces for transmission; fragmented packets must be reassembled when received.

The following sections describe the ICMP techniques the Remote Annex uses to transmit this information.

Not all TCP/IP routers support these features, nor are all hosts configured to take advantage of them. Check the hosts and routers on your network.

Remote Annex Availability – Router Discovery Mechanism

Router Discovery (RD) is an ICMP-based mechanism that helps hosts locate operational routers on directly-connected networks. The Remote Annex uses this mechanism if IP routing is enabled (i.e., if the **routed** parameter is set to **Y**; see *Choosing Passive or Active RIP* on page 9-172).

Using RD, the Remote Annex responds to solicitations from hosts by sending them ICMP messages advertising the Remote Annex as a router. The Remote Annex also broadcasts unsolicited RD advertisements every 10 minutes to all hosts listed on the LAN. If a host does not seem to be learning about Remote Annex routers on directly connected networks, make sure RD is implemented on that host.

RD is not a routing protocol, since it does not help a host decide the best next hop for a particular destination address.

Better First Hops – Redirect Messages

If the Remote Annex determines that there is a better first hop than itself for a datagram it has received, it forwards the datagram and sends an ICMP *redirect* message to the host that originated the datagram. The redirect message contains the address of the next hop the originator should use for subsequent datagrams to that destination.

The Remote Annex itself does not listen for redirects if routing is enabled (i.e., **routed** is set to **Y**). If a redirect is received while routing is enabled, the Remote Annex syslogs the event at WARNING level. If routing is disabled (i.e., **routed** is set to **N**), the Remote Annex listens for ICMP redirects but does not transmit them.

Avoiding Fragmentation – Path MTU Discovery

The Maximum Transmission Unit (MTU) is the largest packet a network can transmit, and it varies depending on the type of network. One way IP accommodates different MTU sizes is by allowing datagrams to be fragmented. Another, more efficient, IP technique avoids fragmentation by discovering the Path MTU, which is the smallest MTU of any hop in the path from a datagram’s source to its destination.

The Remote Annex fragments datagrams when necessary, but also implements Path MTU discovery for datagrams that call for it. If the Don’t Fragment (DF) bit in a datagram’s header indicates the datagram can be fragmented, and the datagram’s size exceeds the MTU of the next hop in the datagram’s route, the Remote Annex fragments the datagram. If the next hop’s MTU is exceeded by a datagram in which the DF bit specifies *not* to fragment, the Remote Annex returns an ICMP *Datagram Too Big* message to the source host. This message includes the next-hop MTU and indicates that fragmentation was needed and the DF bit was 1.

When a host receives a *Datagram Too Big* message, it replaces its estimated Path MTU with the value returned in the message. Typically, the Path MTU discovery process continues until the host succeeds in sending datagrams to a given destination without triggering any *Datagram Too Big* messages.

Routing Interfaces

Initially, RIP runs on all operational IP *interfaces*. Remote Annex interfaces are the SLIP, PPP, and Ethernet ports. A SLIP or PPP interface has a name of the form **asy***n* where *n* is the port number. The Ethernet interface is named **en0**.

The IP address of a Remote Annex interface is the local address you configure for the corresponding port. Typically, the **en0** address is also the IP address of the entire Remote Annex (as defined by the ROM monitor command **addr** or the **inet_addr** parameter).

Interface Routes

For each operational SLIP or PPP link to the Remote Annex that is not on the same subnet (or network, if subnetting is not used) as the Remote Annex, the Remote Annex creates an interface route to the link's remote destination, with the Remote Annex as the next hop. The Remote Annex stores interface routes in the routing table. *These routes are never replaced by routes RIP learns, and you cannot delete them yourself*. (For information on subnetting, see *Subnetting Using Subnet Masks* on page 9-183.)

Non-Operational Interfaces

Except in the case of dial-out interfaces, when an interface goes down, the route for the network attached to that interface is marked as unreachable in the routing table, as are any routes whose next hops are on that network. After two minutes, the Remote Annex deletes the unreachable routes from the table. The CLI command **netstat –i** displays the status of Remote Annex interfaces (see *Interface Statistics* on page B-3).

Dial-out interfaces are treated differently from other interfaces. When a demand-dial or dial-out link has been established and the Remote Annex has received the pertinent RIP routing information, the Remote Annex retains that information even when the link becomes quiescent. After a link becomes quiescent, if an alternate path with an equal metric becomes available for any of the destinations previously reachable over that link, the routes for those destinations are updated to use the new path. However, if you **reset** the link using **na** or **admin**, the routes expire immediately.

IP Addressing

Assigning appropriate IP addresses and subnet masks to your network and the interfaces of attached nodes is essential for routing to work properly. The following sections explain basic IP addressing, subnetting, and supernetting.

An IP address contains four bytes (octets), expressed in decimal, with a period (dot) separating the octets. This is referred to as *dotted decimal notation.* 132.254.1.2 is a sample IP address.

Address Classes

Each IP address contains a network portion and a host portion. To allow for different network-to-host ratios, the Internet supports three classes of IP addresses – A, B, and C – that vary in the number of bytes they allot to the network and host portions. The leftmost byte of an address indicates its class. Table A-7 lists the network classes, the decimal number that appears in the first octet, and the sections of the Internet address that 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-7. 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*

The following values for the first octet are illegal; do not use them: 0, 127 (reserved for loopback), and 224-255.

Sample Addresses for Different Classes

According to Table A-7:

- 125.17.5.2 is a class A address, indicating host 17.5.2 on network 125. Class A addresses allow the most hosts and the fewest networks.
- 132.254.1.1 is a Class B address, indicating host 1.1 on network 132.254. Class B addresses provide the most balanced network/host ratio and are in widespread use on the Internet.
- 192.254.230.6 is a Class C address, indicating host 6 on network 192.254.230. Class C addresses allow the most networks and the fewest hosts.

Example of Class C Address Configuration

Figure A-14 shows a configuration using four Class C node addresses: 194.254.230.1 (*host01)*, 194.254.230.2 (the Remote Annex), 192.252.230.1 (*host02*), and 191.250.230.55 (the PC).

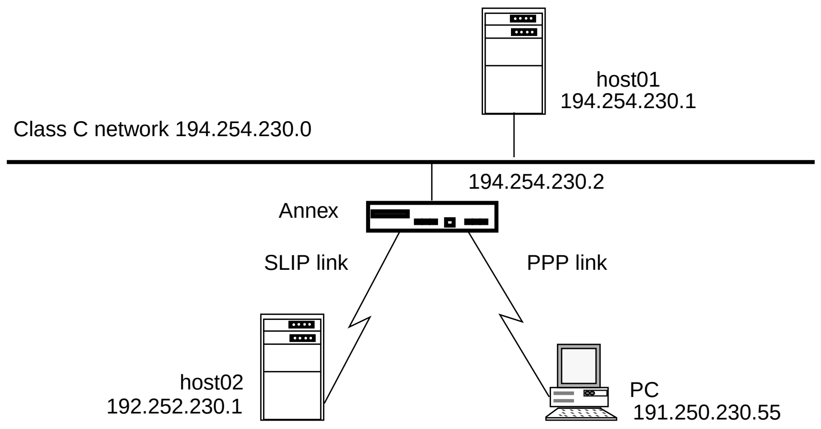


Figure A-14. Configuration Using Four Class C Node Addresses

Obtaining Network Addresses

Before you can assign host addresses, you need one or more network addresses. It is recommended that you use network addresses that can attach to the Internet. Your Internet service provider should supply you with the network address(es) you can use. If you do not have a provider, contact the Registration Services of the InterNIC by sending electronic mail to **hostmaster@rs.internic.net** or by sending U.S. mail to:

Network Solutions
Attn: InterNIC Registration Services
505 Huntmar Park Drive
Herndon, VA 22070

If you have questions concerning registration policy or status, telephone service is available Monday through Friday, 7 a.m. to 7 p.m., Eastern Standard Time. Call 703-742-4777.

Setting the Remote Annex IP Address

Set the Remote Annex address by using the ROM Monitor **addr** command or by setting the configuration parameter **inet_addr**. There are the following special cases:

Turning off IP Services

- Setting **addr** or **inet_addr** to 255.255.255.255 turns off all IP services, including SLIP, PPP, and IP routing. The Remote Annex will continue to support non-IP services, such as ARAP and LAT, provided that they are configured properly. If IP is not being used, turning it off saves overhead and can enhance security.

Enabling IP without an Ethernet Interface

- Setting **addr** or **inet_addr** to a valid IP address and Remote Annex **subnet_mask** to all ones (255.255.255.255) installs IP but leaves the IP address of the Ethernet interface (en0) undefined. IP services, including SLIP, PPP, and IP routing, will be available (if configured properly) but no routes will use the Remote Annex Ethernet interface, if one exists. (If there is no Ethernet interface, you must set the subnet mask to all ones.)

Obtaining the IP Address via BOOTP and RARP

- Setting **addr** or **inet_addr** to 0.0.0.0 causes the Remote Annex ROMs to broadcast for an IP address (at boot time) via BOOTP and RARP. If no IP address is found on the network, the Remote Annex will not boot.

Subnetting Using Subnet Masks

Every IP address on a Remote Annex interface or on a node connected to the Remote Annex has a subnet mask associated with it. You can define the mask yourself or let the Remote Annex assume the default.

If the Remote Annex uses the default mask, your network will not be divided into subnets. The advantages to subnetting a network include:

- Creating a hierarchically organized addressing environment that is logical and easy to remember.
- Reducing the number of entries required in individual routing tables as well as the number of network addresses you must use (and Internet administrators must assign).

A subnet mask lets you designate part of the host portion of an IP address as the subnet. The mask:

- Contains ones in every position that corresponds to the network and subnet part of the address.
- Contains zeros in every position that corresponds to the host address.

For example, used with a Class B address, a subnet mask containing the following bits identifies the first eight bits of the host portion (third octet) as a subnet:

```
11111111 11111111 11111111 00000000
```

For convenience, enter a mask in dotted decimal notation, not in binary. The following is the dotted decimal equivalent of the bits shown above:

```
255.255.255.0
```

You assign a subnet mask to the Remote Annex and to its ports by using the Remote Annex **subnet_mask** parameter and the port **subnet_mask** parameter, respectively. By default, the subnet mask is set to the intrinsic mask for the class of address: 255.0.0.0 for Class A addresses, 255.255.0.0 for Class B addresses, and 255.255.255.0 for Class C addresses. The default mask for a SLIP or PPP port is 0.0.0.0, which is interpreted as 255.255.255.255 (denoting a host route). To use subnetting, you must change these masks.

Incorrectly configuring or not setting subnet masks can cause unrecoverable corruption of the routing table. To detect potential problems, RIP generates a syslog LOG_WARN message if the Remote Annex subnet mask or a port subnet mask is left unset.

The Remote Annex does not accept subnet masks containing non-contiguous one bits.

Figure A-15 shows a simple configuration using subnet addressing. Given a network address of 132.254.0.0, you assign a subnet mask of 255.255.255.0 to *host01* and to the Remote Annex. Assigning this mask defines your network as a subnet whose address is 132.254.1.0 and indicates that the final octet of each remote node defines the host portion of the node's address. In Figure A-15, the host portions are 1, 2, 3, and 4.

For more information on configuring SLIP and PPP ports, see *Serial Line Internet Protocol (SLIP)* on page A-137 and *Point-to-point Protocol (PPP)* on page A-111.

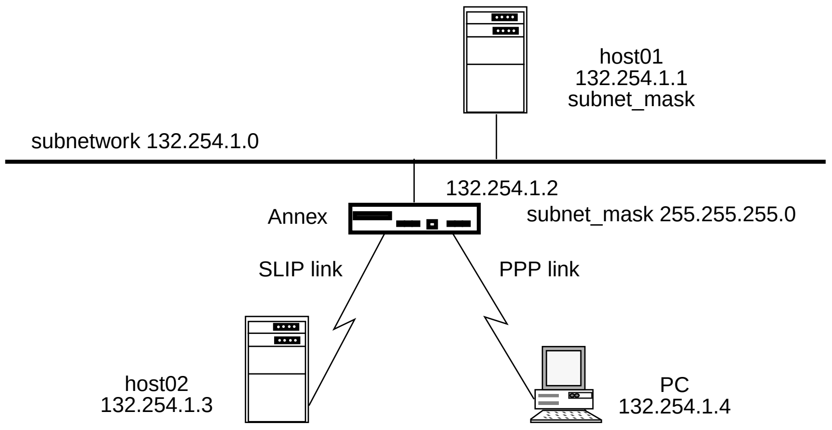


Figure A-15. Subnetting with Passive RIP

Supernetting Using Subnet Masks

The Remote Annex also supports supernetting for Class C addresses. Supernetting allows you to use a subnet mask that is shorter than the intrinsic subnet mask derived from the Internet address' class. Permissible Class C subnet masks range from 255.255.0.0 to 255.255.255.252, excluding 255.255.255.128, which is illegal.

Supernetting Example

Suppose that your Internet Service Provider (ISP) has a block of Class C addresses ranging from 192.24.0.0 to 192.31.255.0. And suppose that from this block, your ISP assigns you 4 Class C network addresses, 192.24.8.0 through 192.24.11.0. To the world outside, the route to your 4-network domain has a destination address of 192.24.8.0 and a mask of 255.255.252.0. Your 4 Class C networks provide you with up to 1024 subnet/host addresses, which you can configure as you wish. For more information on supernetting, see RFC 1519.

Proxy ARP for Interfaces on the Same Network

The section *Interface Routes* on page 9-178 explained that when remote nodes connected directly to the Remote Annex via SLIP or PPP links are not on the same subnet as the Remote Annex, the Remote Annex creates routes for them in its routing table, with the Remote Annex as the next hop. However, when directly-attached nodes *are* on the same or subnet as the Remote Annex (see Figure A-15), the Remote Annex behaves differently. It ignores the interface routes in its tables and uses the Proxy ARP mechanism to forward packets across those links.

Proxy ARP is a variation of the Address Resolution Protocol (ARP), which dynamically maps IP addresses to their physical (Ethernet) addresses. When a network node must resolve an IP address, it broadcasts an ARP request on the network. The node whose IP address matches the one that was broadcast responds with its own Ethernet address. (To display the Remote Annex ARP table, issue the CLI superuser **arp** command; see *arp* on page C-135.)

The Proxy ARP scheme allows the Remote Annex to answer an ARP request on behalf of any directly-attached nodes on its network. In this way, the Remote Annex becomes a Proxy ARP interface for the nodes and installs a static ARP entry for them in the Remote Annex ARP table. When the Remote Annex receives an ARP request for the address of any of these nodes, the Remote Annex automatically responds by providing its own Ethernet address, thereby becoming responsible for forwarding packets to that node. Thus, when the Remote Annex at 132.254.1.2 in Figure A-15 receives a packet for 132.254.1.4, it forwards the packet to the PC at that address.

If one directly-attached node is on the same subnet as the Remote Annex and another is not, the Remote Annex uses Proxy ARP for the former and routing for the latter, as shown in Figure A-16. In this figure, the Remote Annex uses Proxy ARP for the SLIP link on *port1* and routing for the PPP link on *port8*.

Do not attempt to configure a static route whose next-hop address is a Proxy ARP interface. Doing so causes packets to be routed improperly or not routed at all.

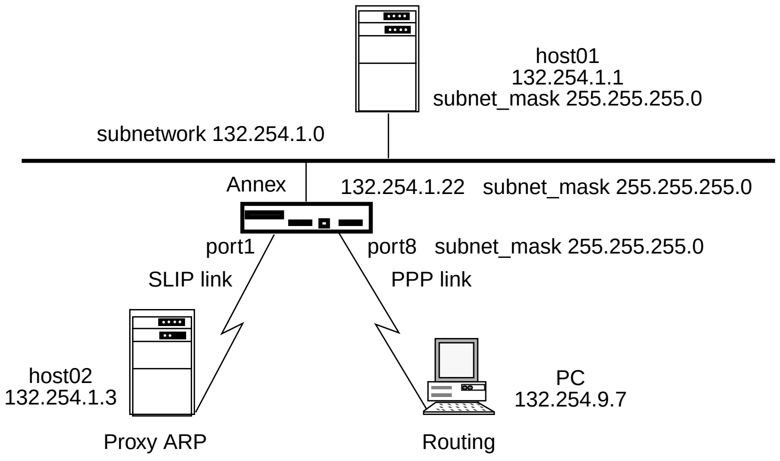


Figure A-16. Proxy ARP versus Routing

To configure *host02* and the PC as in Figure A-16:

1. **On Remote Annex port1, set the** remote_address **parameter to** 132.254.1.3**; the subnet mask setting is irrelevant, since no routing is performed.**
2. **On Remote Annex port8, set the port parameters** remote_address **and** subnet_mask **to** 132.254.9.7 **and** 255.255.255.0**, respectively.**

   Once you have defined the remote end of the link and the subnet mask, the Remote Annex applies the link's local (port) **subnet_mask** to this remote address to determine the route to the remote link.

   You can also set the remote address in the **acp_dialup** file, or, in the case of a PPP link, the Remote Annex can learn the remote address from the remote node itself.

Setting the Broadcast Address

In order for RIP to either receive or advertise routing updates properly, you must set the Remote Annex broadcast address correctly. Use the ROM monitor **addr** command (which prompts you for the broadcast address) or set the Remote Annex **broadcast_addr** configuration parameter.

The IP routing code checks to make sure that the broadcast address is one of the following:

- A subnet broadcast address. If your network is subnetted, this is the recommended broadcast address. To specify this address, set the subnet portion of the broadcast address to match the Remote Annex subnet address, as determined by the Remote Annex subnet mask, and set the host portion of the broadcast address to all one-bits. For example, if the Remote Annex subnet address is 132.254.9.0, and the Remote Annex subnet mask is 255.255.255.252, set the broadcast address to 132.254.9.3. To arrive at this result, subtract the subnet mask from 255.255.255.255. Thus, in the previous example, you subtract 255.255.255.252 from 255.255.255.255 to arrive at 0.0.0.3 and then add this result to the subnet address.
- A network broadcast address. If your network is not subnetted, you can specify this type of broadcast address. To do so, set the network portion of the broadcast address to match the Remote Annex network address, as determined by the intrinsic mask for the network class. And set the host portion of the broadcast address to all 1-bits.
- A limited broadcast address of 255.255.255.255. This reaches all nodes on the subnet. However, if you have more than one subnet on the same physical cable, the Remote Annex will broadcast to all nodes on all of the subnets. This can be troublesome if some of the subnets or nodes do not recognize the broadcast.

If you do not specify one of the broadcast addresses listed above, Remote Annex RIP generates a syslog message. It should be noted that the default for **broadcast _addr** is **0.0.0.0**, which Remote Annex RIP routing does not support (because most hosts do not recognize it).

Overview of Configuration Parameters

This section provides an overview of the RIP-related configuration parameters and the commands for setting them. Subsequent sections describe using these commands. The section entitled *RIP Configuration Parameters – Reference* on page 9-220 provides formal descriptions of the parameters.

RIP configuration parameters fall into two groups – those that apply to the Remote Annex as a whole, and those that apply to particular Remote Annex interfaces (see Table A-8 and Table A-9). Within these two groups, some parameters apply to updates the Remote Annex accepts, and others apply to updates the Remote Annex generates. If only passive RIP is enabled, the parameters you can view and set are limited to the ones controlling the reception of updates. If active RIP is enabled, you can view and set all RIP parameters; the active-only parameters are so noted in the tables.

Table A-8. RIP-specific Remote Annex Parameters

Parameter	Description
rip_auth	Enables and disables authentication of RIP 2 datagrams.
rip_routers	Directs periodic RIP updates to a list of routers rather than broadcasting the updates. *Available with active RIP only.*
routed	Enables and disables RIP.

To display the values of Remote Annex parameters, use the **na** or **admin** command **show annex all**. You can also use SNMP.

The **admin** command **show annex** works the same way as the **na** command **show annex** (see *show* on page C-25). This is true for all the **na** and **admin** commands in this subsection, except for **boot**, an **na** command you cannot issue from **admin.** When using **admin**, return to the CLI superuser command level to invoke **boot** (see *boot* on page C-8).

To set a Remote Annex parameter, use the **na** or **admin** command **set annex** (see *set* on page C-24).

To define several Remote Annexes for subsequent **set** or **show annex** commands to act on as a group, use the **na** command **annex** (see *annex* on page C-6).

Activating Remote Annex Parameter Settings

Reboot the Remote Annex to activate Remote Annex parameter settings.

To display the values of interface parameters, use the **na** or **admin** command **show interface** (see *show* on page C-25).

To set an interface parameter, use the **na** or **admin** command **set interface** (see *set* on page C-24).

To define several interfaces for subsequent **show interface**, use the **na** or **admin** command **interface** (see *set* on page C-24).

Activating Interface Parameter Settings

To activate interface settings, issue the **na** or **admin reset** command – unless you set a parameter for the Ethernet interface (**en0**). In the Ethernet case, reboot the Remote Annex (for more details, see *reset* on page C-21 and *boot* on page C-8).

You can also use SNMP to set and display Remote Annex and interface parameters. The commands vary with SNMP applications. For information on the Remote Annex SNMP implementation, see *Simple Network Management Protocol (SNMP)* on page B-41.

Table A-9. RIP-specific Interface Parameters

Parameter	Description
rip_accept	Controls the networks for which RIP accepts routes and queries.
rip_advertise	Controls the networks for which RIP advertises routes. *Available with active RIP only.*
rip_default_route	Controls whether or not the Remote Annex advertises itself as a default router. *Available with active RIP only.*
rip_horizon	Controls the split horizon and poison reverse mechanisms. *Available with active RIP only.*
rip_next_hop	Specifies whether or not the next hop value is included in RIP version 2 advertisements. *Available with active RIP only.*
rip_accept	Controls the networks for which RIP accepts routes and queries.
rip_advertise	Controls the networks for which RIP advertises routes. *Available with active RIP only.*
rip_horizon	Controls the split horizon and poison reverse mechanisms. *Available with active RIP only.*
rip_next_hop	Specifies whether or not the next hop value is included in RIP version 2 advertisements. *Available with active RIP only.*
rip_recv_version	Controls the version(s) of RIP updates accepted.
rip_send_version	Controls the version(s) of RIP updates advertised. *Available with active RIP only.*
rip_sub_accept	Controls whether or not subnet routes are accepted in updates.
rip_sub_advertise	Controls whether or not RIP advertises subnets. *Available with active RIP only.*

Enabling Passive RIP Alone

The Remote Annex parameter **routed** must be set to **Y** (the default) to enable any type of routing other than static routes. Use **na**, **admin** or SNMP to make sure the setting has not been changed to **N**. If it has been changed, reset it to **Y**, and reboot the Remote Annex (the following example uses **admin**):

```
annex: su
Password:
annex# admin
Annex administration Remote Annex Rx.x, 72 ports
admin: show annex routed

     routed: N

admin: set annex routed Y

     You may need to reset the appropriate port, Annex
     subsystem or reboot the Annex for changes to take
     effect

admin: q
annex# boot
```

At this point, both active and passive RIP are enabled for all interfaces. To use only passive RIP, change the value of the **rip_advertise parameter** from **all** to **none**:

```
admin: interface all
admin: set interface rip_advertise none

     You may need to reset the appropriate port, Annex subsystem or
     reboot the Annex for changes to take effect.

admin: reset interface
```

Configuring Passive RIP

The following subsections describe the default passive RIP configuration and how to change it.

Read this section even if you are using active RIP, since active RIP performs passive RIP as well. The configuration parameters and commands discussed in this section are a subset of those available for active RIP, and descriptions of them are not repeated in the active RIP section that appears later in this chapter.

Defining Routes

Typically, your first configuration task is to define a default route and/or static routes that you believe the Remote Annex will not learn. The Remote Annex provides two ways to do this:

- By entering routes in the Remote Annex configuration file. This method allows you to specify routes that remain defined across Remote Annex boots.
- By issuing the CLI superuser **route** command. This lets you define a static route during a Remote Annex session, thereby avoiding the need to change the configuration file and reboot. However, the route only remains defined for the duration of the session. If the Remote Annex is rebooted, the route disappears.

Entering Routes in the Remote Annex Configuration File

The sections that follow describe how to enter routes in **annex...end** and **subnet...end** blocks in the **gateway** section of the Remote Annex configuration file. Usually, the configuration file path name is **/usr/spool/erpcd/bfs/config.annex** on the load host; you can modify this file using an editor such as *vi*.

You can define routes anywhere in the gateway section of the configuration file, but routes not in an **annex...end** or **subnet...end** block are discarded and not cached if they apply to interfaces that are immediately operational at boot time. Typically, the Ethernet interface is operational immediately, but SLIP and PPP interfaces may be slower to come up.

You may also want to set a default route and/or static routes in the configuration file of one or more hosts on your network. On a Berkeley-style UNIX host, define these routes in the **/etc/gateways** file.

Purpose of a Default Route

The Remote Annex uses its default route when it cannot find a route in the routing table for a particular destination. Initially, no default route is defined. If the Remote Annex receives a default route in a RIP update, it learns and uses it. Or, you can configure a default route using the method described in this section. Once configured, this route is not replaced by any learned routes.

Default Route Entry Format

To define a default route for a Remote Annex, create an entry of the following format in the **gateway** section of the Remote Annex configuration file:

```
%gateway
annex IP_addr
route add [–h] default gateway_address [metric]
end
```

IP_addr is the address of the Remote Annex that will use the default route; *gateway_address* is the IP address the Remote Annex will use as the default gateway; and *metric* is the cost of using this default gateway.

Sample Configuration for Default Routes

Figure A-17 shows a configuration in which Remote Annexes and hosts on the local network must communicate with nodes on a remote network. With only passive RIP enabled on the Remote Annexes, *annex01* cannot reach network *132.254.2.0*, and *host01* and the PC cannot reach either network *132.254.2.0* or *132.254.1.0*. Defining a default route to that network enables the communication.

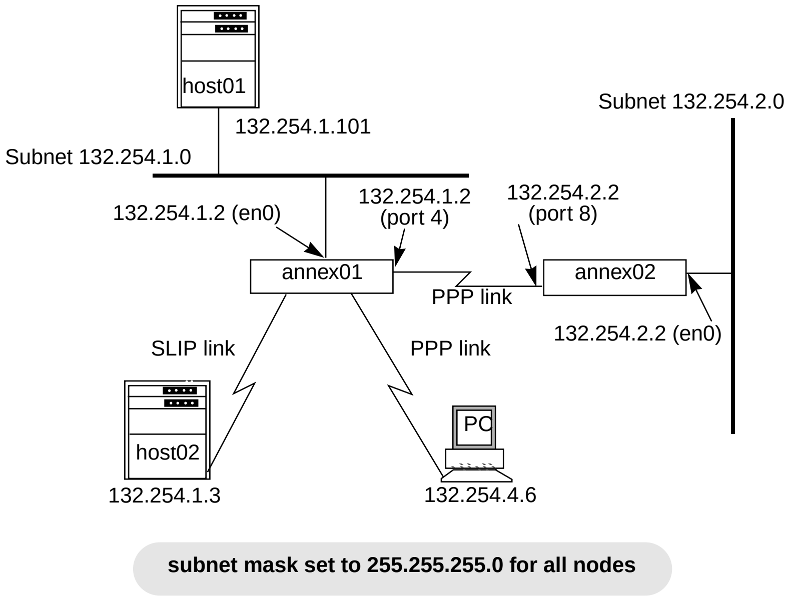


Figure A-17. Sample Network for Defining Default Routes with Passive RIP

For convenience, on the PPP link connecting *annex01* and *annex02*, the local address for each Remote Annex is the same as its **en0** address. You can change this if you want.

Keep in mind that the local address for each Remote Annex is the remote address for the other, e.g., *132.254.1.2* is the remote address of the PPP link from the standpoint of *annex02*.

Entering Default Routes

Use the following procedure to define default routes for the configuration in Figure A-17. This procedure includes setting subnet masks to *255.255.255.0*, as the configuration requires.

1. **On *host01*, *host02*, and the PC, define 132.254.1.2 (*annex01*) as the default route; give it a metric of 1, since it is on a network directly attached to *host1*. The method you use to define the route depends on the host's operating system.**

   Your PC may refer to the default route by another term, such as gateway or router address; check your PC's documentation.

2. **In the configuration file for *annex02,* define 132.254.1.2 as the default route; again, use a metric of 1.**

   ```
   %gateway
   annex 132.254.2.2
   route add default 132.254.1.21
   end
   ```

   Reboot *annex02*, so that it copies the default route into the routing table.

3. **On each host and Remote Annex, and on the PC, define a subnet mask of 255.255.255.0. For a host or PC, the command you use depends on the operating system. For a Remote Annex, use the Remote Annex parameter** subnet_mask**. The following is an example using** admin**:**

   ```
   annex: su
   Password:
   annex# admin
   Annex administration Remote Annex Rx.x, 72 ports
   admin: set annex subnet_mask 255.255.255.0
        You may need to reset the appropriate port, Annex
        subsystem or reboot the Annex for changes to
        take effect.
   admin: q
   annex# boot
   ```

4. **Define 255.255.255.0 as a subnet mask for each end of the PPP link between *annex01* and *annex02* by setting the port parameter** subnet_mask **(see *subnet_mask (port)* on page C-106). Using** admin**, enter the following to set port *asy4* on the *annex01*:**

```
admin: set port=4 subnet_mask 255.255.255.0
       You may need to reset the appropriate port, Annex
       subsystem or reboot the Annex for changes to take
       effect.
admin: reset 4
reset default asynchronous port set [y]
```

5. **Repeat the procedure for *port 8* on *annex02.***

When the PPP link comes up, each Remote Annex adds the remote address for its PPP port as a network route in the routing table. For example, *annex01* adds a route for the *132.254.2.0* network, using *132.254.2.2* as the next hop to that network, with a metric of 1.

If a subnet mask is unset for remote address *132.254.2.2*, the Remote Annex assumes a mask of 0.0.0.0, which is interpreted as 255.255.255.255 (this applies to SLIP and PPP ports only). In this case, *annex01* adds to its routing table a host route for *132.254.2.2*, using *132.254.2.2* as the next hop to that host, with a metric of 1. This host route would be appropriate if *annex02* were not connected to an Ethernet.

Purpose of Static Routes

A static route is one you enter manually, as opposed to a route RIP learns from routing updates. For most network configurations, you typically want to configure at least a few static routes. For example, if a device to which your Remote Annex will be routing does not itself advertise routes, you will want to define a static route to that device. To ensure that the Remote Annex uses defined static routes, specify them in an **annex...end** or **subnet...end** block in the **gateway** section of the configuration file. Static routes defined elsewhere in the **gateway** section are discarded if the interface is not up immediately after the Remote Annex boots. This is not a problem for the Ethernet interface, but SLIP and PPP interfaces can take longer to come up.

annex...end Format for Static Routes

A static route entry in an **annex...end** block has the following format:

```
%gateway
annex annex_IP_addr
route add [-h] dest_IP_addr subnet_mask  gateway_address [metric]
end
```

In the format above:

- *annex_IP_addr* is the address of the Remote Annex that will use the route.
- *dest_IP_addr* is the destination IP address. Do not attempt to give a Proxy ARP host address here; it will not work.
- *subnet_mask* specifies the subnet mask for the destination address. You can enter the mask in dotted decimal notation, e.g., 255.255.255.0, or you can specify the mask by appending a *bits* field to *dest_IP_addr*. Specify the *bits* field as /*n*, where *n* is the number of 1 bits in the mask, from left to right. For example, appending **/24** to an address specifies 255.255.255.0 for the subnet mask.

  Incorrectly configuring or not setting subnet masks can cause unrecoverable corruption of the routing table. To detect potential problems, RIP generates a syslog LOG_WARN message if the Remote Annex subnet mask or a port subnet mask is left unset.

- *gateway_address* is the IP address of the gateway the Remote Annex will use as the next hop to the destination address.
- *metric* is the cost of using this default gateway.
- The **–h** option specifies the route is hardwired, which means that the Remote Annex will not replace the route with any route it learns from RIP.

Table A-10 shows the values you can supply for the *bits* field, along with the resultant subnet mask and its hexadecimal value.

Table A-10. Values for Bits Field with Corresponding Subnet Masks

Bits	Mask	Hex Value	Bits	Mask	Hex Value
8	255.0.0	FF000000	20	255.255.240.0	FFFFF000
10	255.192.0.0	FFC00000	21	255.255.248.0	FFFFF800
11	255.224.0.0	FFE00000	22	255.255.252.0	FFFFFC00
12	255.240.0.0	FFF00000	23	255.255.254.0	FFFFFE00
13	255.248.0.0	FFF80000	24	255.255.255.0	FFFFFF00
14	255.252.0.0	FFFC0000	25	255.255.255.128	FFFFFF80
15	255.254.0.0	FFFE0000	26	255.255.255.192	FFFFFFC0
16	255.255.0.0	FFFF0000	27	255.255.255.224	FFFFFFE0
17	255.255.128.0	FFFF8000	28	255.255.255.240	FFFFFFF0
18	255.255.192.0	FFFFC000	29	255.255.255.248	FFFFFFF8
19	255.255.224.0	FFFFE000	30	255.255.255.252	FFFFFFFC

For each of the valid network classes and subnet bit counts, Table A-11, Table A-12, and Table A-13 show the total number of subnets, and hosts per subnet, that are possible.

Table A-11. Class A: Total Available Subnets and Hosts

Bits	Subnets	Hosts	Bits	Subnets	Hosts
8	1	16,777,214	20	4,094	4,094
10	2	4,194,302	21	8,190	2,046
11	6	2,097,150	22	16,382	1,022
12	14	1,048,574	23	32,766	510
13	30	524,286	24	65,534	254
14	62	262,142	25	131,070	126
15	126	131,070	26	262,142	62
16	254	65,534	27	524,286	30
17	510	32,766	28	1,048,574	14
18	1,022	16,382	29	2,097,150	6
19	2,046	8,190	30	4,194,302	2

Table A-12. Class B: Total Available Subnets and Hosts

Bits	Subnets	Hosts	Bits	Subnets	Hosts
16	1	65,534	24	254	254
18	2	16,382	25	510	126
19	6	8,190	26	1,022	62
20	14	4,094	27	2,046	30
21	30	2,046	28	4,094	14
22	62	1,022	29	8,190	6
23	126	510	30	16,382	2

Table A-13. Class C: Total Available Subnets and Hosts (with no supernetting)

Bits	Subnets	Hosts	Bits	Subnets	Hosts
24	1	254	28	14	14
26	2	62	29	30	6
27	6	30	30	62	2

Sample Configuration for Static Routes

The configuration shown in Figure A-18 requires static (and default) routes. In this example, note the following:

- For convenience, on the PPP link connecting *annex01* and annex*02*, the local address for each Remote Annex is the same as its **en0** address. You can change this if you want.
- The local address for each Remote Annex is the remote address for the other, e.g., *132.254.1.2* is the remote address of the PPP link from the standpoint of *annex02*.
- It is assumed that the SLIP/PPP interfaces on *annex03* are the following networks: 148.254.0.0, 149.12.0.0, 150.14.0, 151.13.0, and 152.254.0.0.

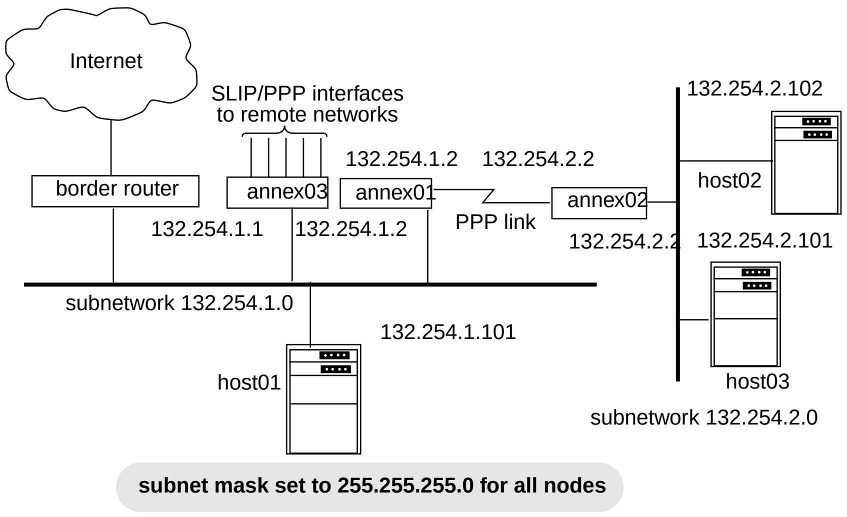


Figure A-18. Sample Network for Static and Default Routes (Passive RIP)

Entering Static Routes

Do as follows to define routes for the network shown in Figure A-18 (a sample gateway section appears after this procedure):

1. **On *host02* and *host03*, define *annex02* (132.254.2.2) as the default route.**
2. **On *annex02*, define *annex01* (132.254.1.2) as the default route.**
3. **On *annex01*, define *border router* as the default route, and configure five static routes defining *annex03* (132.254.1.1) as the next hop for each of the remote networks attached to the *annex03*'s SLIP and PPP interfaces**.
4. **On *host01*, define *border router* as the default route, configure five static routes defining *annex03* (132.254.1.1*)* as the next hop for each of the remote networks attached to the *annex03*'s SLIP and PPP interfaces, and configure *annex01* as the next hop for network 132.254.2.0.**

5. **On *annex03*, define *border router* as the default route, and define a static route to subnetwork 132.254.2.0 via *annex01* (132.254.1.2).**
6. **On *border router*, configure five static routes defining *annex03* (132.254.1.1) as the next hop for each of the remote networks attached to the *annex03*'s SLIP and PPP interfaces. Also, define a static route to network 132.254.2.0 via *annex01* (132.254.1.2).**

To configure static and default routes for the hosts in Figure A-18, use the methods appropriate for the hosts' operating systems. To configure routes for the Remote Annexes in Figure A-18, enter the following **gateway** definition in the Remote Annex's configuration file. Specify static and default routes as **hardwired** if you do not want RIP to replace them with routes it learns over the network.

```
%gateway

annex 132.254.2.2
     #for annex02, use 132.254.1.2 (annex01) as default route
     route add  –h  default  132.254.1.2  1
end
annex 132.254.1.2
     #for annex01, use 132.254.1.1 (annex03) as gateway to 5
     #SLIP/PPP nets
     route add  -h  148.254.0.0/16  132.254.1.1  1
     route add  -h  149.12.0.0/16   132.254.1.1  1
     route add  -h  150.14.0.0/16  132.254.1.1  1
     route add  -h  151.13.0/16   132.254.1.1  1
     route add  -h  152.254.0.0/16  132.254.1.1  1
     #for all other destinations, use 132.254.1.8 (border
     #router) as default
     route add  -h  default  132.254.1.8  1
end
annex 132.254.1.1
     #for annex03, use 132.254.1.2 (annex01) as gateway to
     #132.254.2.0/24 subnet
     route  add  -h 132.254.2.0  132.254.1.2  1
     #for all other destinations, use 132.254.1.8 (border
     #router) as the hardwired default
     route  add  -h  default  132.254.1.8  1
end
```

Routes are automatically added to annex01 for the 132.254.2.0 network, based on the PPP **remote_address** and **subnet_mask** parameters, so no static route needs to be defined for that subnet.

Entering Subnet Routes

You can create **subnet...end** in the gateway section of **config.annex**. This allows you to define a static or default route for all Remote Annexes on a given network or subnet. The syntax is as follows:

subnet *ip_addr*
...
end

The lines enclosed by the **subnet...end** block are to be used only by Remote Annexes on the subnet or network with the IP address *ip_addr*. Any routes enclosed by the **subnet...end** block are cached. An **else** keyword can also be used (alone on a line) to list configuration information for all subnets/networks except the one identified on the **subnet** line. You cannot nest **subnet...end** blocks.

The following are sample **subnet...end** blocks:

```
subnet 132.245.33.0
route add default 132.245.33.22 1
end
subnet 132.245.66.0
route add –h default 132.245.66.22.1
end
```

Entering Routes using the route Command

The CLI superuser **route** command lets you define static routes during a Remote Annex session, thereby avoiding the need to change the configuration file and reboot. The catch is that the route remains defined only for the duration of the session. If the Remote Annex is rebooted, the route disappears.

You can also use **route** to define a default route, but this is not recommended (see *Risks When Adding or Deleting Default Routes* on page 9-209).

The arguments for **route** are shown in Table A-14. The syntax is:

route [**–fF**] **add** [**–h**] *dest mask gateway* [*metric*]

route [**–fF**] **add** [**–h**] **default** *gateway* [*metric*]

route [**–fF**] **delete** {**default** | *dest*}

route [**–fF**] **expire** [**–h**] {**default** | *dest*}

route [**–fF**] **replace** [**–h**] {**default** | *dest*} *gateway* [*metric*]

Specify routes as hardwired (by using the **–h** argument) if you do not want RIP to replace them with routes it learns over the network.

Changes made using the **route** command take effect immediately; you do not need to reboot.

Table A-14. Arguments for the Superuser CLI route Command

Argument	Description
–f	Flushes (deletes) all static and learned routes from the routing table and all static routes from the route cache.
–F	Flushes all hardwired routes from the route cache and routing table. It does not flush interface routes.
–fF	Flushes all non-interface routes from the routing table and route cache.
add	Adds a static route to the route cache. It also adds the route to the routing table if the *gateway* argument (see below) specifies an address that is directly reachable on an active interface.
delete	Deletes a route from the route cache and the routing table. No notification is sent to other routers; they must age the route themselves, which takes 3 minutes. Cannot be used to delete interface routes.
expire	Marks a route as expired by setting its usage to +0 and its metric to 16 (infinity). The lifetime of the route is set to two minutes, during which time RIP continues to advertise the route. This allows other routers to learn that the route is invalid. Cannot be used for interface routes.
replace	Replaces the gateway in the route for *dest* with the new *gateway* you specify. Cannot be used to replace interface routes.
–h	Specifies a hardwired static or default route that RIP cannot replace.

(continued on next page)

Table A-14. Arguments for the Superuser CLI route Command (continued)

Argument	Description
default	Specifies the default route. See *Risks When Adding or Deleting Default Routes* on page 9-209.
dest	Specifies the destination address of the route.
mask	Specifies the subnet mask for the destination address. You can enter the mask in dotted decimal notation, e.g., 255.255.255.0, or you can specify the mask by appending /*n* to the destination address, where *n* is the number of 1 bits in the mask, from left to right. For example, appending /24 specifies 255.255.255.0 as the subnet mask. Table A-10 on page A-200 shows all possible values for the subnet bit count, with the resultant subnet masks.
gateway	Specifies the IP address of the gateway (router) that is the next hop for the route.
metric	Specifies the number of hops to the destination. Values range from **1** through **15**; the default is **1**.

route Command Examples

Both of the following examples add a hardwired route for the destination address 131.108.3. 0 using a subnet mask of 255.255.255.0. This subnet mask indicates that the first two octets are the network address, the third octet (which normally is part of the host portion of this Class B address) is the subnet, and the fourth octet is the host (for more information on subnets, see *Subnetting Using Subnet Masks* on page 9-183). The next hop is 131.254.33.2 and the metric (hop count) for the route is 2. The examples are:

```
annex# route add –h 131.108.3.0 255.255.255.0 131.254.33.2 2
```

and

```
annex# route add –h 131.108.3.0/24 131.254.33.2 2
```

Both of the preceding examples configure the Remote Annex to use the gateway at 131.254.33.2 as the next hop for any host destination whose address is within the range 131.108.3.1 through 131.108.3.254.

Risks When Adding or Deleting Default Routes

Although permissible, using the **route** command to add or delete a default route can have unpredictable results. For example, if you add a non-hardwired default route to the Remote Annex configuration file and then define another one using the superuser CLI **route** command, you have know way of knowing which route the Remote Annex will use. Even if a default route is not in the configuration file, confusion can arise if the Remote Annex learns of a default route through a RIP update. Deleting a default route via the CLI **route** command can cause similar problems. If the Remote Annex knows about two default routes, you cannot know which one you are deleting.

The only time you can safely use the **route** command to add a default route is when one is not defined in the configuration file and the Remote Annex is not receiving RIP updates. In this case, using **route** to define a default route is more convenient than adding the route to the configuration file, since the latter requires rebooting the Remote Annex. Later, you can delete the default route using the **route** command.

Accepting RIP 1 and/or RIP 2 Packets

The Remote Annex's default RIP configuration accepts both version 1 and version 2 packets, making no distinction between version 2 packets that are broadcast and those that are multicast. In most cases, you should not change this default configuration because there may be version 1 RIPs running on your network nodes, and accepting version 2 packets as well does no harm. However, if all nodes on your network run only version 2, you may want to set some or all of the Remote Annex interfaces to accept only version 2 packets. This takes advantage of the fact that RIP 2:

- Supports subnetting better than RIP 1; subnet masks are contained in a RIP 2 route's header, while RIP 1 carries no subnet information.
- When used with the **rip_auth** parameter set to a password, provides full authentication of incoming RIP 2 updates and requests (see *rip_auth* on page 9-223).

To configure some or all of the Remote Annex interfaces to accept only version 2 packets, use **na** or **admin** to set the **rip_recv_version** parameter to **2** for those interfaces. The following example configures all Remote Annex interfaces to accept only version 2 packets:

```
annex: su
Password:
annex# admin
Annex administration Remote Annex Rx.x, 72 ports
admin: set interface=all rip_recv_version 2

     You may need to reset the appropriate port, Annex
     subsystem or reboot the Annex for changes to take effect.

admin: q
annex#: boot
```

The **boot** command is required in the preceding example because all interfaces, including *en0*, are being set. If *en0* were not among the interfaces, you could use the **admin** command **reset interface** instead of the superuser **boot** command.

Do not set **rip_recv_version** to **2** unless you are sure all nodes on your network or internet are advertising only RIP 2 updates.

Authenticating Incoming RIP 2 Updates and Requests

To authenticate incoming RIP 2 messages, set the **rip_auth** parameter to a password containing 1 to 16 characters. The following example sets the password to *ps44D6*.

```
admin: set annex rip_auth ps44D6
admin: set interface=all rip_recv_version 2

     You may need to reset the appropriate port, Annex
     subsystem or reboot the Annex for changes to take effect.

admin: quit
annex# boot
```

RIP 1 packets do not contain a password field, so they cannot be authenticated. Therefore, if you leave the **rip_recv_version** parameter set to the default **both** (which accepts both RIP 1 and RIP 2 packets), setting **rip_auth** provides only partial security.

Once you have set **rip_auth** to a password, an incoming RIP 2 message is authenticated if both of the following conditions are met.

- The message includes the required authentication information. This means that the *Address Family Identifier* field of the first entry in the message must have a value of 0xFFFF, the *Authentication Type* field of the first entry must have a value of **2** (a value of **1** indicates no authentication), and the *Authentication* field must contain a 16-byte unencrypted password.
- The password in the message matches the value of the **rip_auth** parameter.

The Remote Annex accepts all RIP 2 messages it authenticates, but does not necessarily discard all unauthenticated messages it receives. Table A-15 shows how the Remote Annex accepts or discards a RIP message depending on whether or not the Remote Annex and/or the message are configured for authentication and whether or not the password in the message matches the **rip_auth** password.

Table A-15. Remote Annex RIP Version 2 Authentication

	Message Includes Authentication	Message does not Include Authentication
Remote Annex Configured for Authentication	Accepts message if passwords match; otherwise, discards message.	Discards message.
Remote Annex not Configured for Authentication	Discards message.	Accepts message.

Although RIP-2 authentication cannot protect your system against a user who has the physical means or access to diagnostic tools to tap the network, it nevertheless prevents SLIP or PPP users from injecting routes into the system.

Active RIP Prerequisites

The following are prerequisites for using active RIP:

- The **rip_advertise** parameter must be set to **all** for all interfaces and **routed** must be set to **Y**, which are the defaults. To make sure these values are set, issue the following **admin** commands:

  ```
  admin : show interface rip_advertise

  interface en0:
       rip_advertise: all

  interface asy1:
       rip_advertise: all

  interface asy2:
       rip_advertise: all

  interface asy3:
       rip_advertise: all

  interface asy4:
       rip_advertise: all

       .
       .
       .

  admin : show annex routed

       routed: Y
  ```

- At least two Remote Annex interfaces must be operational.
- Each SLIP, PPP, and Ethernet interface over which routing is to occur must have an IP address.
- All SLIP, PPP, and Ethernet interfaces must have appropriate subnet masks. For more information, see *Subnetting Using Subnet Masks* on page 9-183.

Configuring Active RIP

The following section assumes you have read the earlier sections on passive RIP. Where appropriate, this section refers back to the passive RIP sections to avoid repeating material described there.

Defining Routes

With active RIP running, you do not need to define the default and static routes described for the configurations shown in Figure A-17 and Figure A-18. The Remote Annexes will learn about the routes to each other and to other networks through RIP updates they exchange, provided that, for subnetted networks, the **rip_sub_advertise** parameter is set to **Y**, which is the default *(*see *Advertising Subnet Routes* on page 9-216) and that you have configured subnet masks correctly.

Although the routes required for passive RIP need not be defined if you are running active RIP as well, you may want to define a default route and one or more static routes for other purposes. For example, a default router can act as a chokepoint through which all traffic to and from a network must pass. And, you can use static routes to reach routers that are not running active RIP.

To define default and static routes that remain across Remote Annex boots, enter them in the **config.annex** file (see *Entering Routes in the Remote Annex Configuration File* on page 9-194). You can define routes anywhere in the configuration file, but routes not defined in an **annex...end** or **subnet...end** block are discarded and not cached if their interfaces are not operational at boot time. Typically, the Ethernet interface is operational immediately, but SLIP and PPP interfaces may take longer to come up.

To define routes for the current session only, see *Entering Routes using the route Command* on page 9-206.

Advertising RIP 1 and/or RIP 2 Updates

By default, active RIP sends RIP version 2 updates to the IP broadcast address, so that both RIP 1 and RIP 2 systems can receive them. This assumes that **rip_send_version** is set to **compatibility**, which is the default (see *Accepting RIP 1 and/or RIP 2 Packets* on page 9-209 and *rip_recv_version* on page 9-225). It *also* assumes the routers on your network accept both RIP 1 and RIP 2 updates. Although discarding RIP 2 updates violates the RIP 1 RFC (1058), some RIP implementations written before the RFC still do so. If you have both RIP 1 and RIP 2 nodes on your network, make sure that there are no RIP 1 implementations that discard RIP 2 packets. If there are, use **na** or **admin** to set the **rip_send_version** parameter to **1** (see *rip_send_version* on page 9-226), as shown in the following example:

```
annex: su
password:
annex# admin
Annex administration Remote Annex Rx.x, 72 ports
admin: set interface=all rip_send_version 1

      You may need to reset the appropriate port, Annex
      subsystem or reboot the Annex for changes to take effect.

admin: quit
annex# boot
```

The **boot** command is required in the preceding example because you are setting *en0*. If *en0* were not among the interfaces, you could substitute the **admin** command **reset interface** for the **boot** command.

Advertising Subnet Routes

By default, active RIP advertises subnet routes over Remote Annex interfaces. If your network is subnetted, you usually do not want to change this default, unless you are trying to hide your subnet routes from all nodes on a particular interface. In either case, be sure to set the proper subnet masks for the subnet addresses on the network. Also, be aware that RIP 1 packets do not have fields containing subnet masks. Thus, both ends of a RIP 1 communications link must agree beforehand on the part of an address that contains the subnet specification.

Figure A-19 shows an example in which *host03* learns routes to the PC at 132.254.93.2 and the host at 132.254.55.222, even though they are on different subnets from that of the Remote Annex. This is because (1) the Remote Annex has the **rip_sub_advertise** parameter set to **Y**, and (2) all nodes have a subnet mask set (correctly) to 255.255.255.0.

For convenience, the local addresses of the links to the PC and *host14* in Figure A-19 are configured to be the same as the Remote Annex's **en0** interface – *132.254.66.134*.

If *host03* supports host routes, a subnet mask of all 1s (255.255.255.255) would work for the two serial ports on annex01.

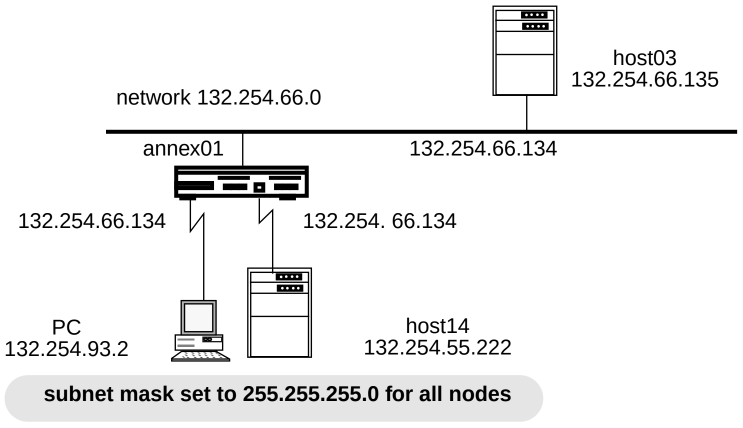


Figure A-19. Advertising Subnet Routes

Subnets are explained in *Subnetting Using Subnet Masks* on page 9-183. The **rip_sub_advertise** parameter is described in more detail in *rip_sub_advertise* on page 9-226.

Using Split Horizon and Poison Reverse

RIP uses a distance-vector algorithm that removes routes from a routing table by aging them and by determining that their destinations are unreachable (more than 15 hops away). To resolve two problems that can occur with distance-vector algorithms, the default Remote Annex implementation of active RIP uses the *split horizon* and *poison reverse* mechanisms.

The split horizon mechanism prevents a route from being advertised as reachable over the interface on which it was learned. Without this mechanism, two routers could advertise the same route back and forth, each adding to the hop count until it reached 16 and the route's destination was determined unreachable. This process could be lengthy and the information conveyed by the hop count, although eventually correct, might no longer be useful.

Poison reverse optimizes the split horizon mechanism. It re-advertises a route over the interface on which it was learned, but does so with a hop count of 16 (unreachable). In general, split horizon with poison reverse is more effective than split horizon alone. If two routers have routes pointing at each other, advertising them as unreachable breaks the loop immediately, while merely not advertising them requires that they time out before being removed from a routing table.

To disable split horizon and/or poison reverse for one or more interfaces, see *rip_horizon* on page 9-224. It is recommended that you set this parameter to either **poison** (the default) or **split**, but not to **off**.

Authenticating Outgoing Updates and Requests

If a Remote Annex is configured for authentication, as described in *Authenticating Incoming RIP 2 Updates and Requests* on page 9-211, the RIP 2 messages it sends to other routers carry authentication information. When the Remote Annex builds the RIP 2 message (update or request), it sets the *Address Family Identifier* and *Authentication Type* fields in the message’s first entry to 0xFFFF and 2, respectively. It also sets the *Authentication* field in the first entry to the value of **rip_auth** (the authentication password configured for the Remote Annex).

RIP 1 updates and requests do not carry authentication information.

Advertising the Default Route

Initially, no default route is defined for the Remote Annex so it cannot advertise one. If RIP receives a default route in an update, the Remote Annex learns, uses, and advertises that route. However, a user-configured default route takes precedence over a learned one, and the Remote Annex does not advertise a user-configured default route.

If you want the Remote Annex to advertise *itself* as a default router over one or more interfaces, set the **rip_default_route** parameter to an integer between 1 and 15 for the specific interfaces. This integer specifies the metric the Remote Annex advertises for itself as a route. Routers on the directly connected network will use the Remote Annex when they have no other route defined for a particular destination.

The Remote Annex need not have a default route itself in order to advertise itself as a default router.

Once the Remote Annex advertises itself as a default route, active RIP no longer advertises a learned default route and instead advertises the route requested by the **rip_default_route** parameter. However, for IP forwarding, the Remote Annex still uses the default route (if any) specified via the **route** command or defined in the configuration file.

Creating or deleting a default route via the **route** command can have unpredictable results and is discouraged except in special circumstances (see *Risks When Adding or Deleting Default Routes* on page 9-209).

Advertising to a Subset of Routers

By default, the Remote Annex sends one update to the broadcast address every 30 seconds so that all routers on the network can receive it. You can restrict updates to only certain routers by specifying them with the Remote Annex parameter **rip_routers**. The routers must be on directly attached networks.

You can specify up to eight routers using the **rip_routers** parameter. The following example specifies two routers.

```
annex: su
password:
annex# admin
MICRO-XL-UX I13.2.21, 16 async, 0 modem ports
admin: set annex rip_routers 132.254.54.2,132.254.54.33
     You may need to reset the appropriate port, Annex
     subsystem or reboot the Annex for changes to take effect.
admin: quit
annex# boot
```

Sending updates to a set of routers can create a significant amount of overhead. For example, if you specify eight routers, eight updates (rather than one) are sent every thirty seconds (one to each router).

RIP Configuration Parameters – Reference

Descriptions of the RIP-specific configuration parameters follow.

Do not change the settings unless you have read the earlier sections of this chapter.

rip_accept

The interface parameter **rip_accept** controls the networks for which routes are accepted from RIP updates and requests. The syntax is:

rip_accept {*access_spec* | **none** | **all**}

Specify *access_spec* as:

{**include** | **exclude**} *network_list*

Use **include** to accept RIP updates only for routes whose destination addresses are on the networks in *network_list*. Use **exclude** to accept all RIP updates except those whose destination addresses are on networks in *network_list*. Do not use **include** and **exclude** within the same command.

For *network_list*, specify up to eight IP network addresses, separated by commas (no spaces are allowed anywhere in the list).

When interpreting an address in *network_list*, RIP uses the intrinsic subnet mask derived from the address class, regardless of the port or the Remote Annex **subnet_mask**
parameter setting.

Instead of *access_spec*, you can enter **none** or **all**; **none** specifies that no RIP messages (updates or requests) are accepted over the interface, while **all** specifies that RIP updates are accepted for all networks. The default is **all**.

The **na** commands in the following example turn off the acceptance of any RIP messages (updates or requests) over interface *asy4:*

```
command: interface asy4
command: set interface rip_accept none

     Changes will take effect at next annex boot or
     interface reset

command: reset interface
```

The **admin** commands in the following example specify that, on interface *asy3*, RIP does not accept routes to destinations on networks 132.254.0.0 and 192.200.0.0. Routes to all other networks are accepted on *asy3*.

```
admin: set interface=asy3 rip_accept exclude\
132.254.0.0,192.200.0.0

     You may need to reset the appropriate port, Annex
     subsystem or reboot the Annex for changes to take effect.

admin: reset interface asy3
```

rip_advertise

The interface parameter **rip_advertise** controls the networks for which routes are advertised. The syntax is:

rip_advertise {*access_spec* | **none** | **all**}

Specify *access_spec* as:

{**include** | **exclude**} *network_list*

Use the keyword **include** to advertise routes for all routes whose destinations are on the networks in *network_list*. Use **exclude** to advertise routes for all whose destination are on networks other than those in *network_list*. Do not use **include** and **exclude** within the same command.

For *network_list*, specify up to eight IP network addresses, separated by commas (no spaces are allowed anywhere in the list).

When interpreting an address in *network_list*, RIP uses the intrinsic subnet mask derived from the address class, regardless of the port or the Remote Annex **subnet_mask** parameter setting.

Instead of *access_spec*, you can enter **none** or **all**; **none** specifies that no RIP updates are advertised over the interface, while **all** specifies that RIP updates are advertised for all network addresses. The default is **all**.

The following **admin** example causes RIP to advertise routes for all destination addresses except those on networks 10.0.0.0 and 190.9.200.0. The example applies to routes advertised over interfaces *asy3* through *asy5*:

```
annex: su
password:
annex# admin
Annex administration Remote Annex Rx.x, 72 ports
admin: interface asy3-5
admin: set interface rip_advertise exclude\
10.0.0.0,190.9.0.0

     You may need to reset the appropriate port, Annex
     subsystem or reboot the Annex for changes to take effect.

admin: reset interface
```

If the above example contained the keyword **include** instead of **exclude**, RIP would advertise *only* those routes whose destination networks were *10.0.0.0* and *190.9.0.0* and no others.

The following **na** example specifies that no RIP messages (updates or requests) are to be sent over interface *asy2*:

```
command: set interface=asy2 rip_advertise none rip_accept\
none

     Changes will take effect at next annex boot or
     interface reset

command: reset interface asy2
```

rip_auth

The Remote Annex parameter **rip_auth** enables and disables RIP 2 authentication. The syntax is:

rip_auth {*password* | ""}

Specifying *password* enables authentication; specifying the null string ("") turns off authentication. The password can be a string of up to 16 characters. The **show annex** command displays this parameter's value as **"<set>"** or **"<unset>"**. The value **"<unset>"** means authentication is turned off; all unauthenticated RIP packets are accepted. The default is the null string (no authentication).

Reboot the Remote Annex to put this parameter setting into effect.

For details on the Remote Annex implementation of authentication, see *Authenticating Incoming RIP 2 Updates and Requests* on page 9-211 and *Authenticating Outgoing Updates and Requests* on page 9-218.

rip_default_route

The interface parameter **rip_default_route** controls whether or not the Remote Annex advertises itself as the default route. The argument *metric* is any value from **0** through **15**, or **off**. A value of **1** through **15** indicates the hop count to be advertised for the Remote Annex's route. A value of **0** or **off** turns off advertisement of the Remote Annex as a default router. The default is **off**. For more information, see *Advertising the Default Route* on page 9-218.

rip_horizon

The interface parameter **rip_horizon** controls the split horizon mechanism: **off** disables split horizon, **split** enables split horizon without poison reverse, and **poison** enables split horizon with poison reverse. The default is **poison**. For more information, see *Using Split Horizon and Poison Reverse* on page 9-217.

rip_next_hop

The interface parameter **rip_next_hop** controls whether or not a route's next hop is advertised in RIP Version 2 updates. Valid options **never**, **needed**, or **always**. The default is **needed**. This parameter has no effect unless the Remote Annex is running more than one IP routing protocol.

rip_recv_version

The interface parameter **rip_recv_version** controls which RIP version the Remote Annex accepts: **1** indicates that version 1 or higher packets are accepted but the non-RIP-1-specific data fields are ignored (i.e., version 2 packets are interpreted as version 1 packets); **2** indicates only version 2 or higher packets are accepted; and, **both** indicates both versions are accepted. The default is **both**. For more information, see *Advertising RIP 1 and/or RIP 2 Updates* on page 9-215.

rip_routers

The Remote Annex parameter **rip_routers** directs periodic RIP updates to a list of routers rather than broadcasting them. The syntax is:

rip_routers *router_list*

The argument *router_list* specifies a list of the IP addresses of up to eight directly reachable routers. Separate the addresses with a comma (and no spaces). The Remote Annex ignores any address not on an attached subnet. Specifying **all** causes RIP updates to be broadcast. The default is **all**.

The following **na** command causes RIP to send updates to the routers at the following addresses: 132.254.33.4, 132.254.1.30, and 132.254.2.2. You must then reboot the Remote Annex.

```
command: set annex rip_routers
132.254.33.4,132.254.1.30,132.254.2.2

     Changes will take effect at next annex boot or
     interface reset

command: boot
```

rip_send_version

The interface parameter **rip_send_version** controls which RIP versions the Remote Annex sends: **1** indicates version 1 packets are sent to the IP broadcast address; **2** indicates version 2 packets are sent to the multicast address; and **compatibility** indicates version 2 packets are sent to the IP broadcast address. The default is **compatibility**. For more information, see *Advertising RIP 1 and/or RIP 2 Updates* on page 9-215.

rip_sub_accept

The interface parameter **rip_sub_accept** controls whether or not subnet routes are accepted. A **Y** accepts subnet routes; an **N** rejects them. The default is **Y**. For more information, see *Advertising Subnet Routes* on page 9-216.

rip_sub_advertise

The interface parameter **rip_sub_advertise** parameter controls whether or not subnet routes are advertised. A **Y** enables subnet advertising; an **N** disables it. The default is **Y**. For more information, see *Advertising Subnet Routes* on page 9-216.

The following **na** command specifies that RIP does not advertise subnet routes over the Remote Annex's Ethernet interface. Note that the Remote Annex must be rebooted to effect the change to **en0**.

```
admin: set interface=en0 rip_sub_advertise n
admin: quit

     You may need to reset the appropriate port, Annex
     subsystem or reboot the Annex for changes to take effect.

annex#: boot
```

routed

The Remote Annex **routed** parameter enables and disables RIP. The syntax is:

routed {**Y** | **N**}

Y enables RIP, **N** disables it. Setting **routed** to **Y** activates both passive and active RIP. The default is **Y**.

Reboot the Remote Annex to put this parameter setting into effect.

Displaying Routing Information

This section describes three diagnostic tools that let you:

- Display RIP interface statistics.
- Display the contents of the Remote Annex routing table.
- Trace the path of a packet as it is routed to its destination.

Displaying RIP Statistics

The **netstat –g** command displays RIP statistics. Table A-16 describes the field definitions for the command display.

The **netstat –g** command display looks like this:

```
annex01# netstat -g
Input packets: 19942, Output packets: 0
Interface triggers: 2, Timer events: 4818   Load trips: 0
Sources:
132.245.33.22:   4661 packets      132.245.33.34:   5632 packets
132.245.33.228:  4822 packets      132.245/33/238:  4816 packets
132.245.33.138:  9                 132.245.33.254:  1 packet

Routing Changes: 1  Queries received: 0

Intf  Bad   Bad   Trigg.  Recv’d  Sent  Disc’d  Update  Queries
      Pkts  Rtes
en0   0     0     0       19942   0     0       22      4
```

Table A-16. Field Definitions for the netstat –g Command

Field	Definition
Intf	Displays the interface.
Bad Pkts	Displays the number of packets the interface dropped due to invalid format or data.
Bad Rtes	Displays the number of routes the interface dropped due to invalid format or data.
Trigg.	Displays the number of triggered updates transmitted over the interface. The Remote Annex sends triggered updates whenever it changes the hop count of a route. It transmits them immediately, even if it is not yet time for one of the regular update messages to be transmitted.

(continued on next page)

Table A-16. Field Definitions for the netstat –g Command (continued)

Field	Definition
Rec’d	Displays the number of packets (with or without errors) received over the interface.
Sent	Displays the number of output packets the Remote Annex tried to send over the interface. This number includes packets that were dropped because the Remote Annex ran out of buffers or the link’s output queue was full.
Disc’d	Displays the number of input packets discarded due to protocol errors or restrictions set by configuration parameters (e.g., rip_accept).
Update	Displays the number on lines in the routing table that were modified due to packets received on that interface.
Queries	Displays the number of routing-table queries received on the interface.

Displaying the Remote Annex Routing Table

To display the contents of the Remote Annex routing table, use the CLI **netstat –r** command, as shown in the example that follows. Note that the command displays AppleTalk routes as well as IP routes. (You can display RIP routes exclusively by issuing the command **netstat –ri**. The IP routes are displayed together, sorted by IP destination address from the lowest to the highest number. The IP fields that **netstat –r** displays are explained in Table A-17.

```
annex: netstat –r
Routing Tables
Destination       NextHop        Flags    Usage    UseCount   Mtr  Interface
32004 - 32005     32005.77       UHF      2        10         0    en0
Apple default     32004.22       UGF      0        0          0    en0
IP default        132.245.66.22  USH      fixed    2          1    en0
1.1.1.1           132.245.66.232 UR       -26760   0          2    en0
1.1.1.2           132.245.66.232 UR       -26760   0          2    en0
1.1.1.3           132.245.66.232 UR       -26760   0          2    en0
1.1.1.4           132.245.66.63  UR       -36      0          2    en0
1.1.1.5           132.245.66.63  UR       -36      0          2    en0
4.0.0.2           *              UI       fixed    0          1    asy2 (ppp)
127.0.0.0/8       *              UI       fixed    0          1    lo0
128.128.0.0/16    132.245.66.66  UR       -26760   0          3    en0
128.128.129.204   132.245.66.122 UR       -18      0          2    en0
131.110.0.8/29    132.245.66.22  UR       -26760   0          2    en0
132.245.1.0/24    132.245.66.22  UR       -26760   0          2    en0
132.245.9.0/24    132.245.66.22  UR       -26687   73         2    en0
132.245.10.0/24   132.245.66.22  UR       -26760   0          2    en0
```

Table A-17. IP Fields in netstat –r Display

Field	Explanation
Destination	The IP address of the route’s destination, followed by a slash (/), followed by the number of 1 bits, counting from left to right, in the Destination’s subnet mask. For example, the */24* following the IP address *132.254.1.0* indicates a subnet mask of 24 bits (eight octets), or 255.255.255.0. (For more information, see *Entering Routes in the Remote Annex Configuration File* on page 9-194.) If *IP Default* appears in the Destination field, the entry specifies the route the Remote Annex uses if it can find no other route for a destination. If a name appears in the Destination field, the entry is for a host route; name servers do not have names for network routes. (However, the Remote Annex does not always know a host’s name.)
NextHop	The next router to which packets with the given Destination are sent. If the Destination is a local interface, this field displays an asterisk (*); interface routes have no next hop.

(continued on next page)

Table A-17. IP Fields in netstat –r Display (continued)

Field	Explanation
Flags	The following three flags:
First flag (Status)	
U	The route is valid (up) and in use.
Q	The route is valid but the interface is quiescent, i.e., the interface is not up yet or was brought down by expiration of the timer set by the **net_inactivity** port parameter.
D	The route is invalid (down) and has a metric of 16 (RIP infinity). It will stay in the routing table for two more minutes so that other routers can learn that it is invalid.
Second flag (Source)	
C	The route was learned via an ICMP redirect. This can occur only when IP routing is disabled (by setting the **routed** parameter to **N**).
I	The route is an interface route.
R	The route was learned via RIP.
S	The route is a static route, learned from a route you defined in the gateway section of the **config.annex** file or a route you entered via the CLI superuser **route** command.
Third flag	
H	The route is a hardwired static route.

(continued on next page)

Table A-17. IP Fields in netstat –r Display (continued)

Field	Explanation
Usage	A positive or negative integer indicating a route's usage. When RIP adds a route to the routing table, it sets its usage value to 0. Every time the route is used, RIP adds 1 to this value. And every thirty seconds, RIP subtracts one from the value. When the routing table reaches its maximum size of 256 entries, RIP removes the route with the lowest usage value. If there is a tie, the first route listed is deleted. The range of values is -9999999, for a route that has not been used in 9.5 years, to +9999999, for a very frequently used route. Interface, hardwired, and *extremely* frequently used routes contain the word *fixed* in this field instead of a number.
UseCount	A positive integer indicating the number of times the route has been used to transmit a packet. If you subtract the value in this field from the value of *Usage*, you can determine how long a route has been in the routing table.
Mtr	The metric associated with the route.
Interface	The interface over which the Remote Annex can reach the next hop.

Displaying the route cache

The **netstat –C** command displays the contents of the cache route, including both static routes added from the **gateways** section of the configuration file and routes added by the **route** command. Table A-18 describes the flags for the command display.

Table A-18. Flag Descriptions for the netstat –C Command

Flag	Definition
intf *x*	An interface route, where *x* is the interface name and number, e.g., asy8. This can be a back-up route for a an interface that has a duplicate definition in the routing table. For example, if you define a subnet mask for a Proxy-ARP serial interface, and that mask is the same as the Remote Annex's en0 subnet mask, the routes to that interface will be considered duplicates. As a result, the Remote Annex will store the (preferred) en0 interface route in the routing table and the serial interface route in the cache, thus making the serial interface unreachable. The example below shows a dial-out route, *do67*.
hardwired	Route added either by the **route –h** command or a route defined as ***hardwired*** in the **gateway** section of the Remote Annex configuration file.

The **netstat –C** command display looks like this:

```
annex01# netstat -C

Destination      Subnet Mask     Gateway        Metric  Flags
default          0.0.0.0         132.245.33.22    1
74.68.67.0       255.255.255.0   0.0.0.0          1     intf do67
132.245.124.0                    132.245.71.72    2     hardwired
```

Using the ping –t (traceroute) Option

The **-t** option of the superuser **ping** command traces the path of a packet from the local host to the destination host and back, displaying information about each router in the path. This option allows you to see whether a packet arrived at and/or returned from its remote destination and, if not, where it stopped. The option is based on the Traceroute facility described in RFC 1393.

The **ping –t** command sends only one ICMP Echo Request. This request, called the *outbound packet*, contains an IP *traceroute* option and a traceroute hop count of zero. If an outbound packet crosses routers on the path to its destination, each router increments the hop count by 1, forwards the packet, if possible, and returns a traceroute message to the originator (Figure A-20 illustrates an outbound packet that crosses two routers). This message indicates whether or not the packet was forwarded. If so, the message contains the incremented hop count and information about the outbound interface over which the packet was forwarded. If the packet could not be forwarded, the router discards it, **ping –t** terminates, and the traceroute message contains zeros in place of interface information.

If an outbound packet reaches its destination, the destination node sends an ICMP Echo Response, called the *return packet*, to the router from which it received the outbound packet. The destination node copies the traceroute option from the outbound packet to the return packet and sets the return packet's hop count to zero. If the return packet passes through the routers in the path back to the **ping –t** source, each router increments the hop count by 1, forwards the packet, if possible, and returns a traceroute message to the **ping –t** source (see Figure A-20).

The traceroute message indicates whether or not the packet was forwarded. If so, the message includes the incremented hop count and information about the interface over which the packet was forwarded. If the packet could not be forwarded, the router discards it, **ping –t** terminates, and the traceroute message contains zeros in place of interface information.

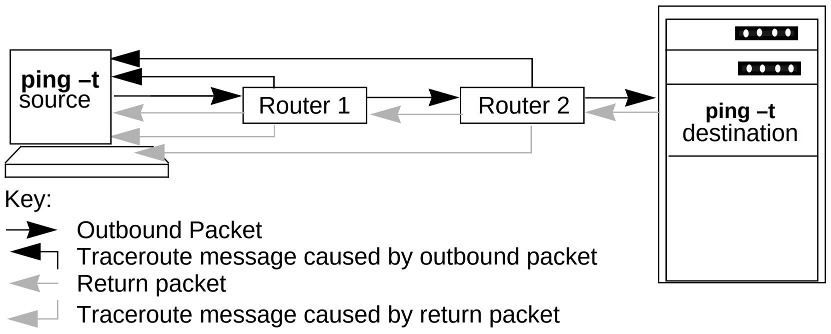

ping –t
source
Router 1
Router 2
ping –t
destination
Key:
Outbound Packet
Traceroute message caused by outbound packet
Return packet
Traceroute message caused by return packet

Figure A-20. Overview of ping –t Actions

Using the information carried in the outbound packet, along with the return packet and the traceroute messages, **ping –t** displays the path of the packets and the characteristics of the routing interfaces along the way and back. And, if a packet cannot be forwarded, **ping –t** locates the failure. Table A-19 on page A-236 describes the fields displayed by **ping –t**.

Table A-19. Fields Displayed by the ping –t Option

Field	Definition
Dir	The direction in which the ICMP packet is heading. The >>> symbols indicate an outbound packet heading towards the **ping –t** destination. The <<< symbols indicate a return packet heading back towards the **ping –t** source. The *** symbols indicate a router could not forward the packet. In this case, the router discards the packet and **ping –t** terminates.
Router	The IP address of the router interface over which the outbound or return packet was forwarded.
Hops	The number of routers that the outbound or return packet has crossed. If the count skips a hop (e.g., goes from 4 to 6), a traceroute message was lost, probably due to network congestion.
Speed	The speed, in bits per second, of the interface over which the outbound or return packet was forwarded. If the packet could not be forwarded, **ping –t** displays a zero in this field.
MTU	The maximum transmission unit (in bytes) of the interface over which the outbound or return packet was forwarded. The MTU is the largest packet size the interface can forward without fragmenting the packet. If the packet cannot be forwarded because its size exceeds the MTU and its header indicates not to fragment, **ping –t** displays a zero in this field.

The sample topology shown in Figure A-21 is assumed by the **ping –t** examples that follow it.

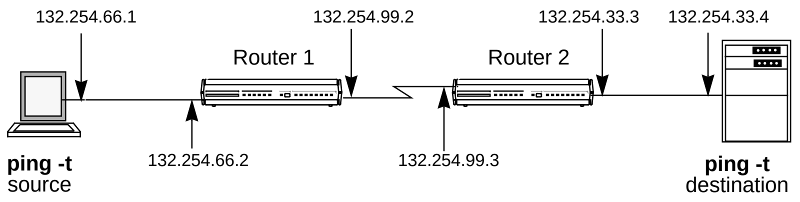


Figure A-21. Topology for ping –t Examples

Given the topology in Figure A-21, the **ping –t** command displays output such as the following when a traceroute packet passes successfully to the **ping –t** destination and back.

The line numbers at the right of this example are for reference only; they are not part of the actual display.

```
annex# ping -t 132.254.33.4
PING hobbes: 56 data bytes                                  line 1

 Dir     Router          Hops   Speed (b/s)    MTU          line 2
 >>>     132.254.99.2    1      19200          1024         line 3
 >>>     132.254.33.3    2      10000000       1500         line 4
 <<<     132.254.99.3    1      19200          1024         line 5
 <<<     132.254.66.2    2      10000000       1500         line 6
 64 bytes from 132.254.33.4: time=10. ms                    line 7
```

In the preceding example:

line 1	Indicates that **ping –t** has started.
line 2	Contains the display header.
line 3	Indicates that Router 1 was the packet's first hop on the path to the **ping –t** destination. The interface over which Router 1 forwarded the outbound packet has an IP address of 132.254.99.2, a speed of 19200 bits per second, and can transmit packets of up to 1024 bytes in length without fragmenting them.
line 4	Indicates that Router 2 was the packet's second hop on the path to the **ping –t** destination. The interface over which Router 1 forwarded the outbound packet has an IP address of 132.254.33.3, a speed of 10000000 bits per second, and can transmit packets of up to 1500 bytes in length without fragmenting them.
line 5	Indicates that Router 2 was the return packet's first hop on the way back to the **ping –t** source. The interface over which Router 2 forwarded the packet has an IP address of 132.254.99.3, a speed of 19200 bits per second, and can transmit packets of up to 1024 bytes in length without having to fragment them.
line 6	Indicates that Router 1was the return packet's second hop on the way back to the **ping –t** source. The interface over which Router 2 forwarded the packet has an IP address of 132.254.66.2, s a speed of 10000000 bits per second, and can transmit packets of up to 1500 bytes in length without having to fragment them.
line 7	Indicates the **ping –t** source has received the return packet and that the round-trip took 10 milliseconds.

In the following example, Router 2 is unable to forward the outbound packet, as indicated by the asterisks (***) under the *Dir* heading. Note that the hop count remains at *1*, since the packet crossed only one router.

```
annex# ping -t 132.254.33.4
PING hobbes: 56 data bytes

Dir     Router          Hops    Speed (B/s)     MTU
>>>     132.254.99.2    1       19200           1024
***     132.254.33.3    1       0               0
```

Troubleshooting

This section describes:

- CLI commands for displaying routing information.
- Common configuration errors.
- What to do if the Remote Annex is not advertising updates as expected.
- What to do if the Remote Annex is not receiving updates as expected.

CLI Commands Providing Routing Information

Use the following commands to obtain information about IP routing on your network.

- To display the contents of the Remote Annex routing table, use the CLI command **netstat –r** (see *Displaying the Remote Annex Routing Table* on page 9-229).
- To display the contents of the routing cache (which contains user-configured routes), use the CLI command **netstat –C** (see *Displaying the route cache* on page 9-233).

Common Configuration Errors

In configuring routing, users make four common mistakes:

- Attempting to route through Proxy ARP interfaces. As has been stated several times in this chapter, do not attempt to configure a static route whose next-hop address is a Proxy ARP interface. Doing so causes packets to be routed improperly or not routed at all.
- Depending on Proxy ARP when routing is more reliable.
- Configuring network/subnet addresses that overlap.
- Configuring non-contiguous subnets.

The following sections describe in detail the final three of these four mistakes.

Depending on Proxy ARP When Routing is More Reliable

Figure A-22 shows a subnet configuration in which a PC can dial into one of two Remote Annexes. Since all nodes are on the same subnet, it is tempting to assume that *host01*could reach the PC via Proxy ARP. However, hosts and Remote Annexes retain Proxy ARP information long enough so that it is possible for the PC to have dialed into one Remote Annex, disconnected, and then dialed into the other Remote Annex before *host01* updated its ARP table. Therefore, the host may not have the up-to-date Proxy ARP information.

To avoid this kind of confusion, assign the PC a different subnet address – e.g., 132.254.6.*x*. This forces *host01* and the Remote Annexes to use routing updates (which are more frequent than ARP requests) to determine the Remote Annex to which the PC is attached. (Make sure that active routing is enabled; see *Configuring Active RIP* on page 9-214. Also, make sure that *host01* supports either subnet or host routes.)

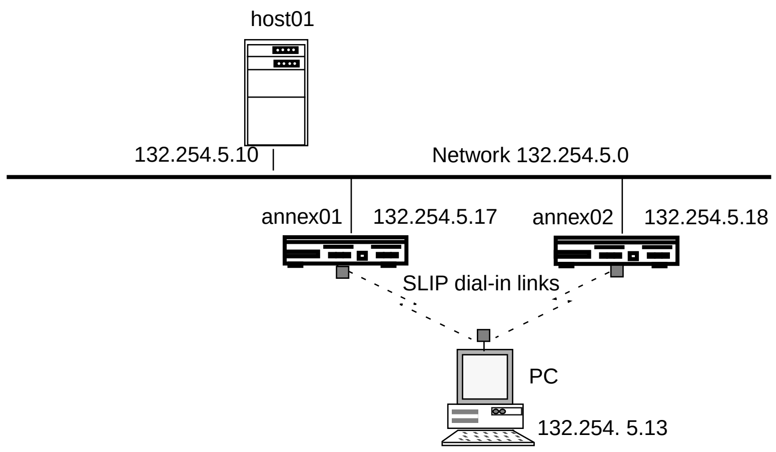


Figure A-22. Configuration in Which Proxy ARP Can Fail

Overlapping Subnet/Network Addresses

Figure A-23 shows a configuration that looks as if it should work but does not. In this configuration, *annex01* is attached to a backbone network and to another annex that is, in turn, connected to a subnet. The problem with the configuration is the subnet mask for *annex01* and *host01*, which designates the two rightmost bytes for host addresses. This mask creates the following kinds of difficulties:

- *Annex01* cannot determine where to send packets addressed to hosts whose addresses fall between 132.254.7.1 and 132.254.7.254. For example, where would it send a packet addressed to 132.254.7.20, which is the address of *host02*? As far as *annex01* is concerned, host 7.20 could be on either of the two networks shown – *132.254.0.0* or *132.254.7.0*.

- Nodes on 132.254.0.0, such as *host01*, cannot reach nodes on subnet 132.254.7.0, such as *host02*. If *host01* wants to send a packet to *host02*, *host01* will try to use ARP to determine the Ethernet address to which it should deliver the packet, since *host02*'s address appears to be on the same directly attached network, *132.254.0.0*. No nodes will respond the ARP request, since *132.254.7.20* is *not* on the local network.

To make the configuration shown in Figure A-23 work, redefine the subnet mask for *annex01* and *host01* as 255.255.255.0. This indicates to the Remote Annex and to *host01* that they are on subnet 132.254.5.0, and that, as a result, *host02* is on a different subnet and must be reached via a router (*annex02*).

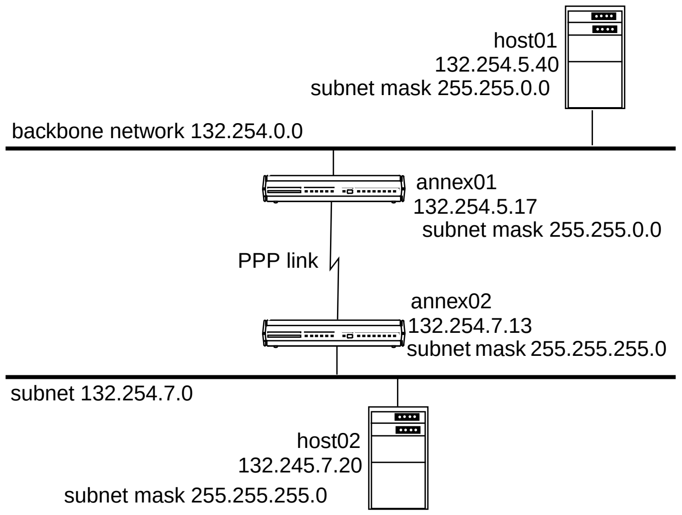


Figure A-23. Overlapping Addresses

Non-contiguous Subnets

Figure A-24 shows a network in which two Remote Annexes are configured to support, via modem pools, two dial-in remote PCs. The problem with this configuration is that both PC's are configured as if they were on subnet 132.254.7.0. Because the subnet masks on the Remote Annex ports are set to 255.255.255.0, both Remote Annexes advertise that they can reach this subnet. Consequently, nodes on 132.254.5.0, such as *host01*, cannot determine which Remote Annex is the appropriate next hop for either PC.

There are two solutions to this problem. The first is to configure both PCs with 132.254.7.*x* addresses to use the same Remote Annex, reserving the other Remote Annex for PCs and/or hosts on a different subnet. The second solution is to configure the ports on the Remote Annexes to use a subnet mask of 0.0.0.0 (the default), which is interpreted as 255.255.255.255, a host subnet mask. This causes the Remote Annexes to advertise host routes, rather than subnet routes. Provided that *host01* and other nodes on subnet 132.254.5.0 accept RIP host route advertisements, they will be able to determine which Remote Annex to use as the next hop for a given PC.

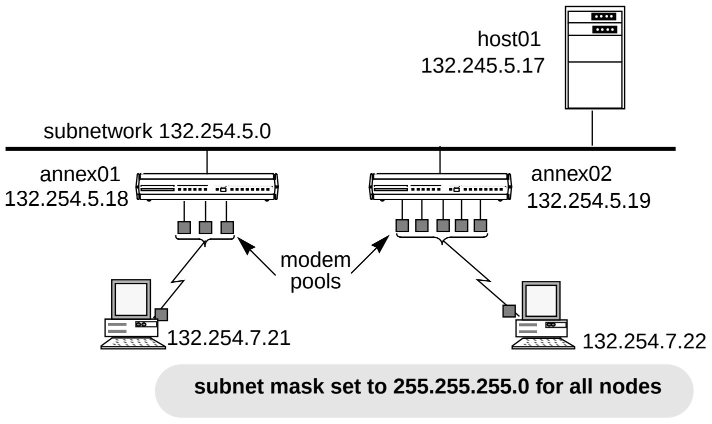


Figure A-24. Non-contiguous Subnets

What to Do if the Remote Annex does not Advertise Updates

If your Remote Annex is not sending RIP updates as expected, check the following:

1. **Is the Remote Annex parameter** routed **set to** Y**? See** ***Configuring Active RIP*** **on page 9-214.**
2. **Did you reboot the Remote Annex after setting** routed**?**
3. **Is the** rip_advertise **parameter set to** all **for all interfaces? To check this, issue the following** admin **commands:**

```
admin : show interface rip_advertise

interface en0:
     rip_advertise: all

interface asy1:
     rip_advertise: all

interface asy2:
     rip_advertise: all

interface asy3:
     rip_advertise: all

interface asy4:
     rip_advertise: all

     .
     .
     .
```

4. **If the display shows that rip_advertise set properly above RIP as enabled, something else is wrong.**
5. **Is the Remote Annex broadcast address set correctly? See** ***Setting the Broadcast Address*** **on page 9-188.**

6. **Are at least two interfaces up and running?**

7. **If your network is divided into subnets, are the IP subnet addresses and subnet masks set correctly for the Remote Annex and the SLIP and PPP ports? See *Subnetting Using Subnet Masks* on page 9-183.**

8. **If your network is divided into subnets, is the interface parameter** rip_sub_advertise **set to** Y **(the default)? See *Advertising Subnet Routes* on page 9-216 and *rip_sub_advertise* on page 9-226.**

9. **Is** rip_horizon **set to** split**? If so, there may not be any routes to advertise on that interface. See *Using Split Horizon and Poison Reverse* on page 9-217 and *rip_horizon* on page 9-224.**

10. **Are RIP packets being filtered out? For example, a filter that discards outgoing UDP packets also discards RIP packets, since RIP runs on UDP. To list all the defined filters, enter the CLI** superuser **command:**

    ```
    annex: su
    password:
    annex# filter list
    ```

    For more information, see *Filtering* on page A-249.

11. **Are your hosts ignoring RIP version 2 updates? If so, set the interface parameter** rip_send_version **to** 1 **(see *Advertising RIP 1 and/or RIP 2 Updates* on page 9-215).**

What to Do if the Remote Annex Does not Receive Updates

If the kernel routing table does not contain the expected learned routes, check the following:

1. **Are the routes really being advertised? Check to see if updates are being received by other routers on the network.**
2. **Did you reboot the Remote Annex after setting** routed**?**
3. **Is** rip_accept **set to** all **(the default)? If not, are the correct network destination addresses being included or excluded via** rip_accept**?**
4. **If your network is divided into subnets, are the IP subnet addresses and subnet masks set correctly for the Remote Annex and the SLIP and PPP ports? See *Subnetting Using Subnet Masks* on page 9-183.**
5. **Is the Remote Annex parameter** routed **set to** Y**? See *Enabling Passive RIP Alone* on page 9-193 and *routed* on page 9-227.**
6. **Is the Remote Annex broadcast address set correctly? See *Setting the Broadcast Address* on page 9-188.**
7. **If subnet routes are not being learned, is** rip_sub_accept **set to** Y **(the default)? See *rip_sub_accept* on page 9-226.**
8. **Is** rip_recv_version **set correctly for the version(s) of RIP running on your network? See *Authenticating Incoming RIP 2 Updates and Requests* on page 9-211 and *rip_auth* on page 9-223.**

Other Documentation

For outside sources of information on TCP/IP, routing, and RIP, see:

- Comer, Douglas E. (1991) *Internetworking with TCP/IP Volume I; Principles, Protocols, and Architecture* (3rd ed.) Englewood Cliffs, N.J., Prentice Hall.
- Hedrick, C.L (1988) *Routing Information Protocol*. RFC 1058 (STD 34).
- Malkin, G. (1993), *RIP Version 2*. RFC 1388.

Chapter 10 Filtering

The Remote Annex implementation of filtering allows you to improve the security of an internal network by preventing potentially dangerous traffic from crossing it. For example, you might want to prevent an outside host from using the Network File System (NFS) protocol or the Trivial File System Protocol (TFTP) to access an internal network, since these protocols have no built-in security and can alter local data. Or, you might want to use filtering to prevent users on your internal network from accessing external hosts and services.

An effective way to provide this kind of protection is to pick one Remote Annex on the internal network to be the network's *chokepoint* or *firewall* through which all traffic to and from external networks must pass. Then, configure filters on that Remote Annex to block undesirable packets (see *add Subcommand Examples* on page 10-264). You can also use filtering to log (in the **syslog** file) traffic for security or network-management purposes. Finally, you can use filters to determine what constitutes traffic on a dial-out serial port.

Filters can apply to one particular physical interface on a Remote Annex or to all Remote Annex interfaces and can affect incoming or outgoing packets. An interface is a SLIP or PPP port named asy*n*, where ***n*** is the port number, or the Ethernet port (*en0*).

Filters are complicated and can interact in ways you might not anticipate; use them with great care.

Filters can cause performance to deteriorate significantly.

Syslogging common occurrences can flood the **syslog** file.

Syslogging *syslogs* can cause infinite loops.

Be careful when creating filters that discard packets on the Ethernet interface; filters of this type can hang the Remote Annex.

You need superuser privileges not only to configure the Remote Annex for filtering but also to create or modify filters.

Include and Exclude

You configure a Remote Annex filter to either **include** or **exclude** particular types of packets, based on whether or not the packet types match specified *criteria*. Including certain types means the filter does not affect any other packet type; excluding certain types means only other types are affected by the filter. The actual effect a filter has depends on the *actions* you specify for it, such as **discard**.

The *criteria* of a single **include** or **exclude** filter are logically ANDed. For an **include** filter, this means that a packet must match all the filter's *criteria* in order for the filter's *actions* to be taken. For an **exclude** filter, it means a packet must match all of the *criteria* for the *actions* ***not*** to be taken.

Multiple **include** filters for the same interface that specify the same *actions* are logically ORed. For example, if one **include** filter for *asy2* specifies that TFTP packets are to be discarded, and another **include** filter for *asy2* specifies that NFS packets are to be discarded, then any packet whose type is either TFTP *or* NFS is discarded; all other packets are accepted on interface *asy2*.

Multiple **exclude** filters for the same interface that specify the same *actions* are logically ANDed. For example, if one **exclude** filter for *asy2* specifies IP address 132.254.45.1 and **discard**, and another **exclude** filter for *asy2* specifies IP address 132.254.55.2 and **discard**, then any packet whose destination address does not match 132.254.45.1 *and* does not match 132.254.45.2 is discarded – i.e., only packets addressed to either 132.245.45.1 or 132.254.55.1 are accepted on interface *asy2*.

In general, use **include** filters to perform an action (such as discard) on only a few types of packets. Use **exclude** filters to exempt only a few types of packets from a particular action.

If both **include** and **exclude** filters are defined for the same interface, the **exclude** filters take precedence. *However,* mixing **include** and **exclude** filters on the same interface is strongly discouraged, since the interactions are subtle and can be confusing even for a networking expert.

Enabling Filtering

Initially, the Remote Annex is not configured to support filtering and the filtering software is not available. To enable filtering on the Remote Annex and make the software accessible, you must enter the correct Remote Annex **option_key** parameter value and reboot the Remote Annex. Each Remote Annex requires a unique **option_key** value; the **option_key** value is case-sensitive.

To enable filtering:

1. **Obtain a valid filtering value for the Remote Annex** option_key **parameter. Each Remote Annex requires a unique value. The way to obtain a key depends on the configuration and type of Remote Annex you purchased. Some option key values are physically attached to the bottom of the Remote Annex. Check there, and enter that value as described in Step 2, below.**

   If there is no option key value attached to your Remote Annex, contact your supplier to obtain a key. You will need to specify the Ethernet address of your Remote Annex; it is taped to the back of the unit.

The **option_key** parameter enables a variety of Remote Annex features, including tn3270, AppleTalk, and IPX, depending upon what you specified when you ordered your Remote Annex.

The filtering **option_key** also enables dial-out.

To determine which options are enabled, issue the CLI **stats –o** command:

```
annex: stats –o

KEYED OPTIONS:

                 LAT: keyed off
               Atalk: keyed off
              tn3270: keyed on
   dialout/filtering: keyed off
                 IPX: keyed off

MODULES DISABLED
     None
```

For filtering to be enabled, *dialout/filtering* must be displayed as *keyed on* and *dialout* must *not* be displayed under the *MODULES DISABLED* heading. For more information, see *disabled_modules* on page C-57.

Although LAT is an option, you set it using the Remote Annex **lat_key** parameter, not **option_key** (see *lat_key* on page C-67).

2. **Use** na, admin**, or SNMP to set the** option_key **value you obtained in Step 1. In this example, the key is set to *RaqbDwv8e* using** admin**:**

```
annex: su
Password:
annex# admin
Annex administration Remote Annex Rx.x, 72 ports
admin: set annex option_key RaqbDwv8e
```

The default superuser password for the Remote Annex is its IP address.

3. **Reboot the Remote Annex to put the filtering** option_key **setting into effect:**

```
admin: q
annex# boot
```

4. **Reconnect to the Remote Annex and issue the CLI** stats -o **command to make sure filtering is Keyed On.**

Accessing the Filter Subcommands

Once you have enabled filtering, use the CLI superuser **filter** command to access the filter subcommands. Invoking this command with no arguments puts you in the filtering subsystem, where you can issue any of the eight subcommands summarized in Table A-20.

The **add** and **delete** subcommands affect both the currently running Remote Annex configuration and the configuration stored in non-volatile memory. Rebooting the Remote Annex or issuing a **reset** command is not necessary.

You cannot access the **filter** command from **na**, and you cannot save the filter parameters to a file using the **na** command **write**

The following example shows entering the filtering subsystem and issuing the **list** subcommand to display the current filters:

```
annex# filter
filter: list

 Num  Stat  Ifname Dir  Scope Family Actions/Parameters
 1    ena   en0    in   incl  ip     disc icmp/port_pair=*,nfs
 2    ena   en0    in   incl  ip     disc/port_pair=*,tftp
 filter:
```

To return to the superuser CLI from the filter subsystem, use the filter subcommand **quit**:

```
filter: quit
annex#
```

You can also issue **filter** subcommands directly at the CLI superuser prompt by using the syntax:

filter *subcommand*

The following shows the **list** subcommand issued from the CLI superuser prompt. When **list** completes, you return to the CLI superuser prompt.

```
annex# filter list

 Num  Stat  Ifname Dir  Scope Family  Actions/Parameters
 1    ena   en0    in   incl  ip      disc icmp/port_pair=
                                      *,nfs
 2    ena   en0    in   incl  ip      disc/port_pair=*,tftp
 annex#
```

Table A-20. Summary of filter Subcommands

Subcommand	Description
add	Adds filters and automatically enables them.
delete	Deletes filters.
disable	Disables filters but does not delete them.
enable	Enables filters.
help	Displays a one-line description of one or all **filter** subcommands.
list	Displays filters.
quit	Exits the filtering subsystem, returning control to the CLI.
usage	Displays the syntax for one or all **filter** subcommands.

Filter Numbers

When you **add** a filter, the Remote Annex assigns it a number that remains associated with it until you delete the filter. The **filter** subcommand **list** displays this number, and you specify the number when you **delete**, **enable**, or **disable** a filter (see *Filter Lists* below for permissible ways to specify filters).

Filter Lists

The **delete**, **enable**, and **disable** subcommands accept a *filter_list* argument. A filter list can be a filter number, a string of filter numbers separated by commas, or a range of filter numbers. Use a dash (**–**) to separate the beginning and end of a range, or use it before or after a filter number. Used before a filter number, a dash indicates all defined filters up to and including that number; used after a filter number, a dash indicates all filters from that number up to and including the highest numbered filter.

Specifying an asterisk (*), the word **all**, or a dash (**–**) by itself for a filter list indicates all filters. Table A-21 shows sample subcommands using filter lists.

When you delete filters, their numbers remain unused until you add another filter; the added filter is then assigned the lowest unused number. If you invoke a subcommand with a range that includes unused numbers, the subcommand operates on the assigned filters but displays an error message for each unused number. This does not happen when you specify a group of filters by entering a number with a leading or trailing dash; in this case, unused numbers are ignored.

Table A-21. Sample Commands using the filter_list Arguments

Argument	Description
delete 2	Deletes filter 2. A subsequent **list** subcommand will not display an entry for filter number 2.
disable 3–6	Disables filters 3, 4, 5, and 6. If one of these numbers represents a deleted filter or an existing filter associated with an inactive interface, an error message is displayed for that number; the other filters in the range are disabled.
disable 1, 3–7,10	Disables filters 1, 3, 4, 5, 6, 7, and 10. If any of these numbers represents a deleted filter or a filter for an inactive interface, an error message is displayed for that number; the other filters in the range are disabled.
disable –5	Disables filters 1, 2, 3, 4, and 5.
disable 3–	Disables all filters from filter 3 through the end of the list of all filters.
enable –	Enables all filters.
enable *	Enables all filters.
enable all	Enables all filters.
enable 5–	Enables all filters from filter 5 through the end of the list of all filters.

Filter Subcommands

This section describes the subcommands in alphabetical order.

add

The **add** subcommand adds new filter(s) and enables them in both the currently running system and non-volatile memory; the Remote Annex need not be rebooted for the added filters to take effect. Table A-22 describes the arguments for **add**. The syntax is:

add *interface direction scope* [*family*] *criteria actions*

Table A-22. Arguments for the add Subcommand

Argument	Description
interface	Specifies the physical interface to which this filter applies. Valid values are **en0** (for Ethernet) or **asy***n*, where *n* is a SLIP or PPP port number. Specifying “*****” applies the filter to all interfaces.
direction	Specifying **input** applies the filter to incoming packets. Specifying **output** applies the filter to outgoing packets. Two filter definitions are required to apply a filter to both incoming and outgoing packets.
scope	Specifying **include** means the filter matches only those packets that meet all of the specified *criteria*. Specifying **exclude** means the filter matches only those packets that do not meet any of the specified criteria.
family	(Optional) Specifies the network level address family (protocol) to which the filter applies. Currently, the Remote Annex only supports **ip**.
criteria	Specify the conditions on which the filter is based. All criteria must be met for the filter to match the packet. Specify criteria in the form: **keyword** *value* (see Table A-23).
actions	Specifies what a filter does when all of its *criteria* match a packet. You can specify any combination of *actions*. Possible *actions* are:

(continued on next page)

Table A-22. Arguments for the add Subcommand (continued)

Argument	Description
discard	Discards the packet. Discarding is done after any **syslog**, **icmp**, or **netact** actions are taken.
icmp	Discards the packet and sends an ICMP *destination unreachable* message.
netact	Used only with dynamic dial-out networking. Customizes the definition of activity for a SLIP or PPP dynamic dial-out line. If one or more filters containing this action are enabled on one of these line, only the traffic matching the filters constitutes activity (see *Dial-up Networking* on page A-155 for more information on dynamic dialing). If the link is quiescent, **netact** discards the packet.
no_start	Used only with dynamic dial-out networking. If used in an include filter, the **no_start** filter action specifies that traffic defined as activity on the specified interface will not activate a dynamic dial-out line. However, the filter action will keep the line up if it is already up–that is, if it is not quiescent. In the process, the action resets the **net_inactivity** timer to 0. See *Dial-up Networking* on page A-155 for information on dynamic dialing.
syslog	Logs the event in the system log file.

Filters that would cause multiple **syslog**, **icmp**, and **netact** *actions* for the same interface are reduced to a single **syslog**, **icmp**, or **netact** action.

To add a dynamic dial-out filter, configure the **dialout** section of the Remote Annex configuration file; you cannot add the filter using the **filter** command (see *Creating dialout Entries in the Configuration File* on page A-386).

Table A-23 lists valid keywords and values for the **add** subcommand's *criteria* argument. The syntax is:

keyword value

Table A-23. Keywords for add criteria Argument

Keyword	Value	Explanation
dst_address	{*ip_addr*[/*n*] | ***** | **-1**}	Matches the packet's destination IP address. To test only the non-host portion of the address, enter /*n* after the address, where *n* is the number of bits in the non-host portion of the subnet mask for this address. For example, **132.245.33.0/24** denotes a mask of 255.255.255.0, which matches destination addresses on network 132.245.33.0. (If you list the filter, 132.245.33.0/24 appears as the destination address.) To match all addresses, enter **–1** or "*****" instead of an address.
dst_port	{*pnum* | *sname* |***** | **-1**}	Matches the TCP or UDP destination port. Specify the port as a decimal number (*pnum*) from **1** – **65535** or as a standard service name (*sname*), such as **finger**, **ftp**, **nfs**, **rlogin**, **smtp**, **telnet**, or **tftp**. Specifying **–1** or "*****" matches all port numbers. For a list of service names and their corresponding port numbers, see Table A-24.

(continued on next page)

Table A-23. Keywords for add criteria Argument (continued)

Keyword	Value	Explanation
src_port	{*pnum* | *sname* | ***** | **-1**}	Matches the TCP or UDP source port number. Specify the port as a decimal number (*pnum*) from **1** – **65535** or as a standard service name (*sname*), such as **finger**, **ftp**, **nfs**, **rlogin**, **smtp**, **telnet**, or **tftp**. Specifying **–1** or “*****” matches all port numbers. For a list of service names and their corresponding port numbers, see Table A-24.
src_address	{*ip_addr*[/*n*] | ***** | **-1**}	Matches the packet’s source IP address. To match only the non-host portion of the address, enter /*n* after the address, where *n* is the number of bits in the non-host portion of the subnet mask for this address. For example, **132.245.33.0/24** denotes a mask of 255.255.255.0, which matches destination addresses on network 132.245.33.0. (If you list the filter, 132.245.33.0/24 appears as the destination address.) To match all addresses, enter **–1** or “*****” instead of an address.

(continued on next page)

Table A-23. Keywords for add criteria Argument (continued)

Keyword	Value	Explanation

address_pair	{*ip_addr1*[/*n*] | ***** | **-1**}
	
	{*ip_addr2*[/*n*] | ***** | **-1**}
	
	(Enter both addresses on the same line; separate them with a space)
		Matches packets passing in either direction between two specified IP addresses. To match only the non-host portion of an address, enter /*n* after the address, where *n* is the number of bits in the non-host portion of the subnet mask for this address. For example, **132.245.33.0/24** denotes a mask of 255.255.255.0, which matches destination addresses on network 132.245.33.0. (If you list the filter, 132.245.33.0/24 appears as the destination address.)
		
		To match all packets to or from a given address, enter one *ip_addr* and then specify ***** or **-1** for the other. For example, *** 132.254.33.2** and **132.254.33.2 -1** match all packets to or from 132.254.33.2.
		
		Restriction: if you use the address_pair keyword, you cannot use the keyword dst_address or src_address.

(continued on next page)

Table A-23. Keywords for add criteria Argument (continued)

Keyword	Value	Explanation
port_pair	{*p1 p2*| *s1 s2* |*****|**-1**}	Matches packets passing in either direction between the two specified TCP or UDP port numbers (*p1* and *p2*) or standard service names (*s1* and *s2*), such as **finger**, **ftp**, **nfs**, **rlogin**, **smtp**, **telnet**, or **tftp**. Use a space to separate the port numbers or names. To match all packets to or from a given port number, enter one port number or service name and specify **–1** or ***** for the other. For a list of service names and their corresponding port numbers, see Table A-24. Restriction: if you use the port_pair keyword, you cannot use the dst_port or src_port keyword.
protocol	{*protonum*|*protoname*}	Matches the transport protocol in the packet. Valid protocol numbers range from **1** to **65535**. Or, specify a protocol name, such as **tcp**, **udp**, or **icmp**. If no protocol is given but a port is specified (dst_port, src_port, or port_pair), the port specification applies to both TCP and UDP packets. **Warning**: A command such as the following can cause infinite loops: **filter add asy1 output include\** > **protocol icmp icmp**

Table A-24 shows the standard service names and port numbers you can supply for service name and port number values in Table A-23.

Table A-24. Standard Service Names and Corresponding Port Numbers

Service Name	Port Number
domain	53
finger	79
ftp	21
name	42
nfs	2049
nntp	119
rlogin	221
route, routed, router	520
rtelnet	107
sftp	115
smtp, mail	25
snmp	161
telnet	23
tftp	69
time	37
who, login	513

Multiple service names shown on the same line in Table A-24 are synonyms. Using any one of them in a filter implies using the other. However, when you list the filter using the **list** subcommand, you will see only the first service name.

add Subcommand Examples

Since the NFS and TFTP protocols do not support password protection, you may want to use filtering to prevent hosts on an external network from using those protocols to access files on your internal network. To do this, pick a Remote Annex to act as a firewall between the local and external network and create filters on it to block NFS and TFTP traffic. For example, you could create the following two filters, which prevent TFTP or NFS packets from crossing Remote Annex interface *asy8*:

```
annex# filter
filter: add asy8 input include protocol udp port_pair\
> nfs * icmp
filter: add asy8 input include protocol udp port_pair\
> tftp * icmp
```

Note the following about the preceding sample filters:

- Both filters apply only to packets arriving on Remote Annex interface *asy8* (which could attach to either a PPP or SLIP line). To apply a filter to an another interface, specify a second filter for that interface, or specify ***** instead of *asy8*, thereby blocking the protocol on all interfaces.
- Both filters match packets whose network protocol family is IP. Since the **family** argument is optional (IP is assumed), the examples omit it.
- Both filters specify **protocol** as UDP because UDP is the transport-level protocol on which NFS and TFTP operate.
- The **port_pair** argument in each filter specifies that the filter applies to any UDP packet that contains NFS or TFTP in its source or destination protocol field.

- When one of these filters matches a packet, the Remote Annex discards the packet and sends the ICMP message *destination unreachable, communication administratively prohibited* to the originator of packet. To discard the packet without sending a message, specify **discard** instead of **icmp**.

The following example creates a filter that logs the arrival of every IP packet on the Ethernet interface (*en0*). The example omits the network protocol family because it is optional; IP is assumed.

```
filter: add en0 input include src_addr * syslog
```

Logging events requires configuration (see *Logging User and Annex Events* on page B-26 for more information).

The following example allows packets to and from 132.254.100.2 and 132.254.100.3 to be forwarded over interface *asy1*; all other packets are discarded.

```
filter: add asy1 input exclude address_pair 132.254.100.2\
> * discard
filter: add asy1 input exclude address_pair 132.254.100.3\
> * discard
```

The following example allows UDP and ICMP packets to and from 132.254.100.2 and 132.2534.100.3 to be forwarded over interface asy1; all other packets are discarded.

```
filter: add asy1 input exclude address_pair 132.254.100.2 *\
> protocol icmp discard
filter: add asy1 input exclude address_pair 132.254.100.3 *\
> protocol icmp discard
filter: add asy1 input exclude address_pair 132.254.100.2 *\
> protocol udp discard
filter: add asy1 input exclude address_pair 132.254.100.3 *\
> protocol udp discard
```

delete

The **delete** subcommand deletes filters from both the currently running system and non-volatile memory. The syntax is:

delete *filter_list*

See *Filter Lists* on page 10-255 for an explanation of the *filter_list* argument.

Dial-out filters can be deleted but will be re-installed after a reboot of the Remote Annex or a reset of the dial-out subsystem.

disable

The **disable** subcommand disables enabled filters in the currently running system. The filters remain defined in non-volatile memory, but packets are not matched against them. When you reboot your Remote Annex, the disabled filters are re-enabled. The syntax is:

disable *filter_list*

The following example disables filters 2 through 4:

```
annex# filter
filter: disable 2-4
filter 2 disabled
filter 3 disabled
filter 4 disabled
```

See *Filter Lists* on page 10-255 for an explanation of the *filter_list* argument.

Dial-out filters can be disabled but will be re-enabled after a reboot of the Remote Annex or a reset of the dial-out subsystem.

enable

The **enable** subcommand immediately enables disabled filters in the currently running system. Otherwise, disabled filters are not enabled until the Remote Annex reboots or until the port resets. A filter can be disabled only by the **disable** command. The syntax is:

enable *filter_list*

See *Filter Lists* on page 10-255 for an explanation of the *filter_list* argument.

The following example enables filters 2 through 4:

```
annex# filter
filter: enable 2–4
filter 2 enabled
filter 3 enabled
filter 4 enabled
```

help

The **help** subcommand displays information about one or more **filter** subcommands. Entering **help** with no arguments displays information about all of the subcommands. Entering **help** and the name of a subcommand displays an explanation of that subcommand. For a one-line syntax, see *usage* on page 10-270.

list

The **list** subcommand displays the defined filters, along with their status (enabled or disabled) and assigned number (used by **enable**, **disable**, and **delete**). Table A-25 describes the arguments for **list**. The syntax is:

list [**–eia**]

Table A-25. Arguments for the list Command

Argument	Description
–e	Lists the filters stored in non-volatile memory instead of the filters in the currently running system. Using **list –e** eliminates the status column from the display because the enabled/disabled status is not saved in non-volatile memory.
–i	Sorts the output by interface name, instead of sorting by filter number.
–a	Sorts the output by action, instead of sorting by filter number.

If you do not specify **–e**, **list** displays only those filters that are associated with active interfaces.

The **list** subcommand display looks like this:

```
annex# filter
filter: list

 Num  Stat  Ifname  Dir  Scope  Family Actions/Parameters
 1    Ena   en0     in   incl   ip     disc icmp/port_pair=
                                       nfs,*
 2    Dis   all     out  incl   ip     syslog/addr_pair=
                                       132.254.31.2,*
```

In the preceding example, two filters are displayed. The first filter:

- Is enabled.
- Applies to the inbound Ethernet interface.
- Includes all IP packets coming from or destined for the TCP/UDP NFS port.
- Discards the packet.
- Sends an ICMP *destination unreachable* message to the packet’s originator.

The second filter:

- Is disabled.
- Applies to all outbound interfaces;
- Includes all IP packets coming from, or destined for, the host with an IP address of 132.254.31.2.
- Generates a *syslog* message.

quit

The **quit** subcommand exits the filtering subsystem and returns control to the CLI.

usage

The **usage** subcommand displays the syntax for one or more **filter** subcommands. Entering **usage** with no arguments prints syntaxes for all the subcommands. Entering **usage** and the name of a subcommand prints the syntax for that subcommand.

For more detailed information about one or more subcommands, see *help* on page 10-267.

Chapter 11
Internetwork Packet Exchange (IPX) Protocol

IPX is the network-layer communications protocol Novell networks use to deliver data packets to their destinations. The Annex implementation provides Novell dial-in connectivity and routing for asynchronous serial ports.

Novell Networks

Nodes on Novell network are *servers* or *clients*. Servers provide shared access to files, printers, and specialized peripheral devices on the network. Within this context, the Annex functions as a communications server, providing shared access to the network by non-Novell as well as Novell nodes. *Clients*, also referred to as *workstations*, connect to the server(s) via a network interface (Ethernet, Token Ring, or Arcnet) to access files and services. The most common client and server hardware platforms are PCs.

The Novell environment is unlike that of UNIX, in which users connected by terminals execute programs on a UNIX host. In the Novell environment, users generally execute programs on the client, not on the file server. The programs are stored on the server and retrieved from there for execution on the client. Moreover, a client PC, unlike a terminal, can operate as a stand-alone computer, since it has its own processor, storage, operating system, and application software.

IPXCP Features

The Annex implements standards-based IPX (IPX over PPP) via the IPX Control Protocol (IPXCP) described in RFC 1552. IPXCP allows a PC to dial into an Annex port as an endpoint node on an IPX network. The same PC can also simultaneously run IP over the connection, allowing the user to use either IP or IPX services as the need arises. (The same link can also be used for AppleTalk over PPP.)

To dial into an Annex via IPXCP, a PC client can be running any operating system that supports IPXCP networking. This includes Windows '95, Windows NT, and DOS or Windows running FastLink II version 2.*x* or higher.

In addition to dial-in access, IPX over PPP provides asynchronous routing, allowing an IPX network to be run across a PPP LAN-to-LAN link. The Annex provides this type of routing by default: an Annex configured for IPXCP automatically sends and accepts RIP and SAP packets, provided that you set a network number for the link (see *Configuring IPXCP Routing* on page 11-288). However, a client can choose whether or not to receive RIPs and SAPs (if the client software allows this choice).

Annex IPXCP does not support NLSP routing.

Figure A-25 shows a sample configuration in which a remote client (*PC1*) can connect to an Annex (*annex02*) via IPXCP. This allows *PC1* to access the services of the Remote Office network as if directly attached to it. Moreover, by using the LAN-to-LAN IPXCP link, the machines in the Central Office (Servers 1 through 4 and annex01's UNIX boot host) can access the nodes in the Remote Office, including *PC1*, and vice versa. *annex01* and *annex02* must be connected directly, from PPP port to PPP port, via a hardwired null-modem line or a leased line.

.

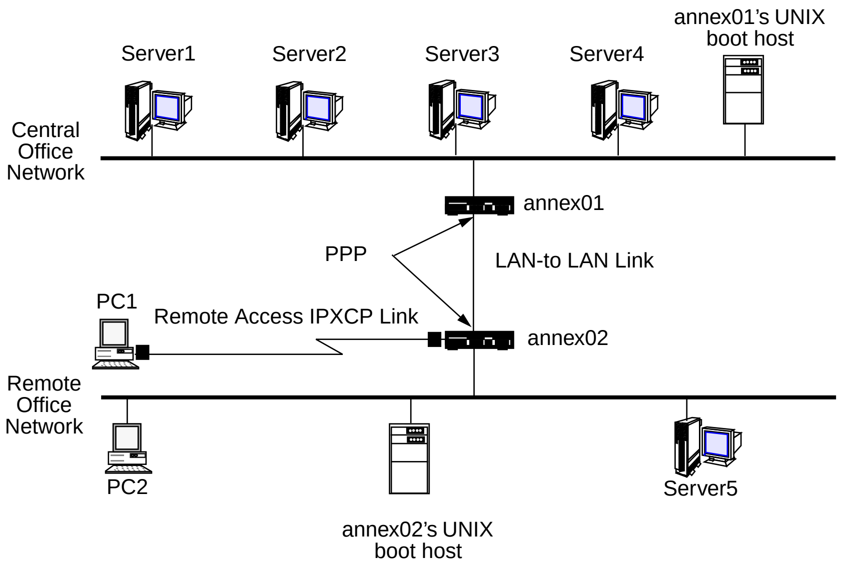


Figure A-25. Sample IPXCP Network Configuration

To configure the Annex for IPXCP, see *Configuring IPXCP* on page 11-276.

The current Annex implementation of IPXCP does not support dial-out, call-back, or charge-back, except for ACP dial-back on a CLI port.

Enabling IPXCP

Initially, all IPX functions are disabled on a Remote Annex. To enable IPXCP, set the **option_key** parameter. You can set the **ipx_frame_type** parameter at the same time to avoid having to reboot twice. The procedure that follows explains setting both of these configuration parameters.

1. **Obtain a valid IPX value for the Annex** option_key **parameter.**

   Each Remote Annex requires a unique option key. The way to obtain a key depends on the configuration and type of Annex you purchased. Some option key values are physically attached to the bottom of the Annex. Check there, and enter that value as described in Step 2, below.

   If there is no option key value attached to your Annex, contact your supplier to obtain a key. You will need to specify the Ethernet address of your Annex; it is taped to the back of the unit.

   The **option_key** parameter activates a variety of Annex features, including tn3270, AppleTalk, dialout and filtering, and IPXCP. When requesting an IPXCP **option_key**, mention any other **option_key** features currently enabled for your Annex.

To determine which options are activated (*Keyed On*), issue the CLI **stats** command with the **–o** option, as follows:

```
annex: stats –o

KEYED OPTIONS:

 LAT:                    keyed off
 Atalk:                  keyed off
 tn3270:                 keyed on
 dialout/filtering:      keyed off
 IPX:                    keyed off

MODULES DISABLED
     vci
```

The *DISABLED MODULES* field displays any software modules that have been disabled via the Annex **disabled_modules** parameter (see *disabled_modules* on page C-57). If this field displays *ipx*, then IPX is unavailable even if the IPX **option_key** parameter is set correctly.

2. **Set the** option_key **parameter to the value you obtained in Step 1.**

In the following example, the option key is set to *RaqbDwv8e* using **admin**:

```
annex: su
Password:
annex# admin
Annex administration Remote Annex Rx.x, 32 async ports
admin: set annex option_key RaqbDwv8e
```

The **option_key** value is case-sensitive.

The default superuser password for the Annex is its IP address.

3. **Specify the type of IPX frame that the Novell network uses to encapsulate IPX packets on the Ethernet.**

   To specify the frame type, set the Annex **ipx_frame_type** parameter. Valid values are **ethernetII**, **raw802_3** (the default), **802_2**, or **802_2snap**. The following command sets the type to **802_2**:

   ```
   admin: set annex ipx_frame_type 802_2
   ```

   To determine the frame type the network is using, check the AUTOEXEC.NCF file on your Novell server or use the Novell file server console command called PROTOCOL.

4. **Reboot the Annex to put the parameter settings into effect:**

   ```
   admin: q
   annex# boot
   ```

5. **Reconnect to the Annex and issue the CLI** stats –o **command to make sure that IPX is *Keyed On* and that *ipx* is not listed as a disabled module.**

6. **Issue the** stats **command with no arguments and check to make sure the network number is not displayed as *0*.**

   A network number of 0 indicates a misconfigured **ipx_frame_type** parameter or the absence of an IPX server on the network.

To disable IPX, set the Annex **disabled_modules** parameter to **ipx** and reboot the Annex. The **stats –o** command should now display *ipx* in the *DISABLED MODULES* field (see Step 1).

Configuring IPXCP

The following sections describe:

- Configuring IPXCP dial-in.
- Configuring IPXCP routing.

Configuring IPXCP Dial-in

You can configure IPXCP for specific users or for specific Remote Annex ports. Configuring for specific users allows you to deny or permit Annex IPXCP dial-in access to a subset of clients. Configuring on a per-port basis limits the Annex ports that can accept dial-in IPXCP packets. The sections that follow describe:

- The minimum IPXCP configuration required to establish a connection between an Annex port and an IPXCP client. This configuration uses the default settings for IPXCP and other parameters and does not provide security.
- More extensive IPXCP configuration for the following.
  - Specific users. This involves using the **acp_dialup** file on a UNIX host on your network. Configuring security is covered briefly in this section. For complete information on security, see *Using PPP Security* on page A-514.
  - Specific ports. This involves setting the port parameters **ppp_ipx_network** and **ppp_ipx_node**. For complete information, see *Using PPP Security* on page A-514.

Minimum Configuration for IPXCP Ports

Assuming the configuration in Figure A-26, and also assuming that you have not changed any configuration parameter defaults, you can establish an IPXCP link between the PC and port 4 on the Remote Annex 4000 by using the following procedure:

1. **Enable IPXCP as described in *Enabling IPXCP* on page 11-274.**
2. **Set the** mode **parameter to** ppp **for port 4.**
3. **Set the** speed **parameter for port 4 to the highest rate at which the modem connected to the port can communicate.**

4. **Set the** type_of_modem **parameter to the type of modem connected to port 4.**
5. **To prevent characters from being dropped between the modem and the Annex, configure port 4 for hardware (EIA) flow control.**

   To configure EIA hardware flow control, set the **control_lines** port parameter to **both** and the **input_flow_control** and **output_flow_control** port parameters to **eia**.
6. **Reboot the Annex (to activate the** option_key **setting, recalculate the IPX buffer pool, etc.).**
7. **Dial into the Annex from the PC to test the connection.**

Keep in mind that this simple procedure does not provide Annex security, among other things. For more IPXCP configuration possibilities, see the next section.

The above procedure does not assign a network and node number to the remote client, but instead uses the values suggested by the client. If the client does not suggest values, the Remote Annex uses its own Ethernet address plus 1for the node address and a randomly generated number for the network address.

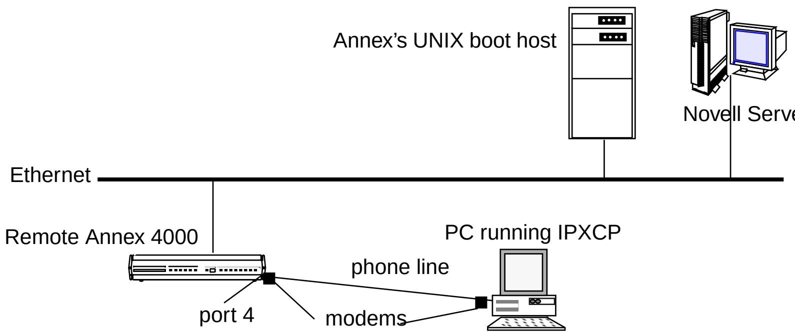


Figure A-26. Sample Hardware for Minimum IPXCP Configuration

IPXCP Configuration via the acp_dialup File

Figure A-27 illustrates a configuration using the **acp_dialup** file. In this configuration, a single PC connected to an Annex through a IPXCP link appears to the network as a directly attached node. Following the figure are the parameter settings required for this configuration. These include security settings. However, for complete information on configuring security for IPXCP, see *Using PPP Security* on page A-514; IPXCP uses PPP security.

Setting up an Annex port for dial-in IPXCP requires that you configure the port for both an inbound modem and a IPXCP link (see *Modems* on page A-99 for more details on inbound modems).

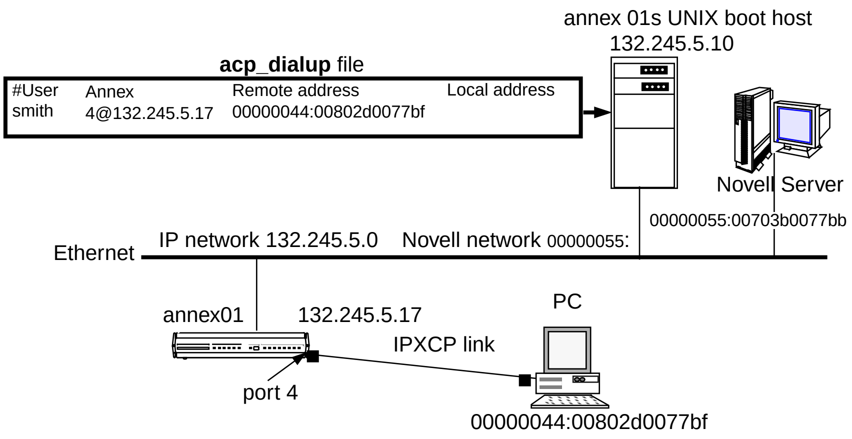


Figure A-27. Connecting a Single Host Using IPXCP

1. **For *port 4* on *annex01*, set the** mode **parameter to** ppp**,** cli**,** auto_detect**, or** auto_adapt**.**

   - Use **ppp** mode if the remote client (the PC) expects to dial into a port that is already running IPX over PPP.
   - Use **cli** mode if you are using third-party security software, such as SecurID. Once having met the configured security requirements, the user issues the **ppp** command to set the mode to **ppp**.
   - Use **auto_detect** or **auto_adapt** mode if you want the port to detect any of the following protocols: IP over PPP, IPX over PPP, and AppleTalk over PPP. This mode also allows the port to detect PPP, ARAP and the CLI. (SLIP is not detected.)

   If you set a port to **auto_detect** or **auto_adapt** and you want the Annex to be secure, configure the port to use native protocol security for every protocol type the port could detect. This means configuring the port for PPP, IPX, CLI, and AppleTalk security.

2. **Establish the Network Control Protocol (NCP) to be negotiated for *port 4*.**

   The Annex supports the following NCPs: AppleTalk Control Protocol (ATCP), Internet Packet Exchange Protocol Control Protocol (IPXCP), Internet Protocol Control Program (IPCP), CCP (Compression Control Protocol for PPP links), and Multilink PPP (MP). NCP options are negotiated in the same way as LCP options. An NCP peer opens the link and the interface is available to the Annex (for information on Multilink PPP, see the *Multilink PPP Addendum to the Remote Annex Administrator's Guide for UNIX)*.

   To specify one or more NCPs, set the **ppp_ncp** port parameter to any combination of **ipxcp**, **ipcp**, **atcp**, **mp**, and **ccp**. Separate multiple values with a commas. You can also specify **all** to indicate all of the protocols, which is the default.

If you specify **ccp** as an NCP, the Annex automatically requests data compression for a PPP link. Three types of compression are negotiated:

- Predictor-1, a public-domain algorithm
- BSD-Compress, a freely-available portion of the BSD UNIX sources
- STAC (with Check Modes 1, 3, and 4), a licensed, standard compression scheme. Check Mode 4 is one of the compression types used by the Windows '95 Dial-up Networking feature.

These three compression types have higher compression ratios than that provided by V.42 bis in standard modems.

3. **Set the port** type **parameter to** dial_in **if you want to register the user with the who database as soon as a CLI process attaches to the line.**
4. **Turn security on for the for the Annex by setting the Annex** enable_security **parameter to** Y**.**
5. **Define annex01's UNIX boot host as the host used for security by setting the Annex** pref_secure1_host **to** 132.245.5.10**.**
6. **The** slip_ppp_security **parameter controls dial-in PPP access. If** enable_security **and** slip_ppp_security **are set to** Y**, access to the ppp command is controlled via ACP and port access is logged in the ACP log file.**
7. **Set the port parameter** ppp_security_protocol **to** pap**, or** chap-pap.**, or** none**.**

8. **To enable CLI and/or connection security, set the port security parameters:** cli_security**,** connect_security**, and** port_password**.**

9. **Set the port parameter** allow_compression **to** Y **if you want the Annex to accept and transmit compressed packets. For IPXCP links, the Annex uses CIPX header compression.**

10. **Use the supplied defaults for the port parameters** data_bits **(**8**),** stop_bits **(**1**), and** parity **(**none**).**

PPP is an 8-bit protocol. If **data_bits** is set to **7**, and **parity** is not set to **none**, the Annex forces the **data_bits** setting to **8** and the **parity** setting to **none**. If this is not possible because **data_bits** is set to **7** and **parity** is set to **none**, the Annex *syslogs* an error message for the port.

11. **Set the** type_of_modem **parameter to the type of modem attached to the port.**

12. **Set the** speed **parameter to the highest rate at which the modem on the port can communicate.**

13. **To configure the port for hardware (EIA) flow control, set the** control_lines **parameter to** both **and the** input_flow_control **and the** output_flow_control **parameters to** eia**.**

14. **To configure the port for software (XON/XOFF) flow control (not recommended) instead of hardware flow control, set the** control_lines **parameter to** none **and the** input_flow_control **and** output_flow_control **parameters to** start/stop**. If you do this, the Annex automatically modifies the** ppp_acm **parameter appropriately (see *ppp_acm* on page C-87).**

15. **Configure the** acp_dialup **file in the install directory on the UNIX boot host (the default directory is** /usr/annex**).**

    In **acp_dialup**'s *user* field, specify the user name that will be associated with the dial-up address request (*smith* in the sample configuration). In the *annex* field, enter the IP address or host name of the Annex the user will be dialing into (*132.245.5.17* in the sample configuration). Leave the *Local address* blank (the Annex uses its own Ethernet address for the node portion of the local address). For the *Remote address*, enter the (unique) Novell address of the remote PC. Use the form *network***:***node*, as the following describes.

    network is a string of 8 hexadecimal digits representing the 4-byte, non-zero Novell network number of the remote PC. Valid values are 00000001 to FFFFFFFF. Include leading zeroes, if any. You can also specify an asterisk for network, as explained below. Specifying zero (0) for network is a syntax error; the Annex ignores the entry. A network number must be unique from other network numbers on the network and the Annex itself.

    node is a string of 12 hexadecimal digits representing the 6-byte, non-zero node number of the remote PC. Valid values are 00-00-00-00-00-01 to FF-FF-FF-FF-FF-FE (make sure you enter the dashes), except for multicast addresses. A multicast address is any address that has a 1 in the last bit of the first octet. For example, the Appletalk multicast address is 090007000000, of which the first octet (09) is 0000 1001 in binary; the rightmost 1 is the multicast indicator.

    When you specify *node*, include any leading zeroes However, specifying zero (0) alone for *node* is a syntax error; the Annex ignores the entry.

    You can also specify an asterisk (*) for *node*, as explained below.

    In parsing the **acp_dialup** file, ACP uses a first-match algorithm keyed on the format of the *Remote address* field, as well as on the contents of the *user* and *annex* fields. That is, if an entry matches the user name and Annex address, but the Remote address is not of the form *network:node*, no match is found. In this case, the Remote Annex uses the values, if any, set for the port parameters **ppp_ipx_node** and **ppp_ipx_network** *(*see *IPXCP Configuration for Specific Ports* on page 11-285).

If *network:node* is configured properly in **acp_dialup** and the **ppp_ipx_network** and **ppp_ipx_node** are also set, the **acp_dialup** values are used and the port parameters are ignored. If valid values are not set in **acp_dialup** or via the port parameters, the Remote Annex uses the values suggested by the remote client. If no values are suggested by the client, the Remote Annex uses its own Ethernet address plus one for the remote node address and uses a randomly generated number for the network address.

If you specify an asterisk (*) for either the network or node portion of the remote address, the PC is allowed to choose the network or node number during NCP negotiation. If you set both portions of the address to asterisk (*:*), the remote client is allowed to choose both the network number and the node number. If you specify * for both portions of the address and the remote client does not try to negotiate an address, the Remote Annex:

–Sets *node* to the Annex's own Ethernet address plus 1.

–Sets *network* to a unique, randomly generated value.

Unless you specify an asterisk for the network number, the Annex always uses the number you specify in **acp_dialup**, even if the PC client tries to negotiate a higher number. This is not compliant with the IPXCP RFC 1552, which states that the highest number should be used.

However, giving control of the network number to the Annex provides better security. The only problem is that if the PC client suggests a higher number during NCP negotiation, the client may refuse the Annex number and the IPXCP connection will not be established.

If this is a concern for you, configure the IPXCP network number via the **ppp_ipx_number** parameter, which always uses the highest value

16. **Set the Annex parameter** enable_security **to** Y **so that the Annex uses the** acp_dialup **file and any other security features you configure.**

17. **Set the** address_origin **parameter to** acp **so that the Annex requests the endpoint addresses from the** acp_dialup **file.**

    If **address_origin** is not set to **acp**, or if it is set to **acp** but no IPXCP addresses of the proper form are defined in **acp_dialup**, the Remote Annex uses the address defined by the **ppp_ipx_network** and **ppp_ipx_node** parameters.

18. **You can leave the** ppp_mru **parameter set to its default.**

19. **Reboot the Annex (to recalculate the IPX buffer pool).**

At this point, user *smith* should be able to dial into the *port 4*. If the port mode is **cli**, *smith* must pass CLI security and then issue the **ppp** command at the CLI prompt. If the port mode is **auto_detect** or **auto_adapt**, *smith* presses **Return** to enter **cli mode** and *then* issues the **ppp** command after passing CLI security. The Annex PPP asks the security server (132.245.5.10) for *smith's* address. Then the Annex negotiates with *smith's* PC for this address and opens the link.

Windows '95 IPXCP clients must make sure that SPAP security is not enabled on their PCs. SPAP is a proprietary Microsoft security mechanism not supported by many other systems, including the Remote Annex.

IPXCP Configuration for Specific Ports

To configure IPXCP on strictly per-port basis, follow the previous instructions for configuring IPXCP for specific users, but substitute the following for steps 15 and 17

15. **Instead of configuring the** acp_dialup **file, set the port parameters** ppp_ipx_network **and** ppp_ipx_node**.**

Set **ppp_ipx_network** to any valid Novell network number. This network number is a string of 8 hexadecimal digits representing the 4-byte, Novell network number of the remote PC. Valid values are 00000001 to FFFFFFFF, or 0. Include leading zeroes, if any. The network number must be unique on the network and on the Annex itself. When the IPXCP connection is established, the Annex and the client negotiate, each suggesting a value for the network number. The peer suggesting the highest number wins the negotiation, and the network number is set to that value. If both ends of the link set the network number to 0, a unique, randomly-generated number is used.

Set **ppp_ipx_node** to a string of 12 hexadecimal digits representing the 6-byte, non-zero node number of the remote PC. Valid values are 00-00-00-00-00-00 to FF-FF-FF-FF-FF-FE, except for multicast addresses. A multicast address is any address that has a 1 in the last bit of the first octet. For example, the Appletalk multicast address is 09-00-07-00-00-00, of which the first octet (09) is 0000 1001 in binary; the rightmost 1 is the multicast indicator. If the client suggests any valid value for the node number, that number will be used instead of the **ppp_ipx_node** value.

17. **Set the** address_origin **parameter to** local **instead of** acp**, so that the Annex will not look for endpoint addresses in the** acp_dialup **file.**

    Remember that values in the **acp_dialup** file override the **ppp_ipx_network** and **ppp_ipx_node** parameters. If the node number is not set in **acp_dialup** or through the **ppp_ipx_node** parameter, and no value is suggested by the client, the Annex uses its own Ethernet address plus 1.

Configuring IPXCP Routing

To configure IPXCP LAN-to-LAN routing between two Annexes:

1. **Set a port to PPP mode on each Annex.**
2. **Attach a null modem or leased line to each of the PPP ports.**
3. **Use the** ppp_ipx_network **parameter to set the network number for at least one end of the PP link. For valid network numbers, see *IPXCP Configuration for Specific Ports* on page 11-285.**
4. **Issue the CLI command** reset annex all**.**

Accessing Network Resources via Fastlink II

Version 2.13 of Fastlink II allows a user to dial in using a standard telephone line to a Remote Annex running PPP protocol. Once connected, the user becomes a node on the network with IP and IPX access to all network resources. Included in Version 2.13 is the option to install the Novell LAN Workplace TCP/IP stack. Included with the Novell TCP/IP stack are the utilities **PING.EXE** and **TRACERT.EXE**.

Fastlink II supports the following IP stacks:

- Beame & Whiteside (BW-TCP 3.2)
- Frontier (Super/TCP 4.0 R2)
- FTP (OnNet v2.1 for Windows)

- Microsoft (MS TCP/IP 3.11b for WFW 3.11)
- NetManage (Chameleon 4.01)
- Novell (LAN Workplace for DOS/Windows 4.2, 5.0)
- Wall Data (Rumba for Internet 1.0)
- Wollogong (Pathway Access 3.1)
- WRQ (Reflection 2, 4.01)

Configuring IPXCP Routing

To configure IPXCP LAN-to-LAN routing between two Annexes:

1. **Set a port to PPP mode on each Annex.**
2. **Attach a null modem or leased line to each of the PPP ports.**
3. **Issue the CLI command** reset annex all**.**

Obtaining IPX Information

IPX information is available from several sources, including log messages the Annex creates automatically and output that various commands display.

System Logs

The Annex automatically logs **auto_detect/auto_adapt**, and **ppp** port events to a 4.3BSD system log daemon (**syslogd**) or to a serial port on the Annex. To send log messages to a port, use the Annex parameter **syslog_port**. For more information on syslogging, see *Using Event Logging* on page A-37 and *Logging Security Events* on page A-545.

IPXCP Interface Statistics

The **netstat –ip** command displays the IPXCP state and IPXCP options. The following is an example of IPXCP statistics for port 6:

```
annex: netstat -ip 6

 *** LCP Status ***
 State                  Current: Open      Prior: Ack sent
 Options                Local:             Remote:
 MRU                    1500               1500
 Auth type              None               None
 LQM                    None               None
 ACFC                   On                 On
 ACCM                   0x00000000         0x000a0000
 Magic                  0xbb1ee499         0x0047501b
 PFC                    On                 On


 *** NCP(IPXCP)Status ***
 State                  Current: Open      Prior: Ack sent
 Options                Local:             Remote:
 Network No             12345678           12345678
 Node No                00802d009c30       00802d009c30
 Compression            None               None
 Routing Prot           RIP/SAP            RIP/SAP
 Router Name            LM009c30           None
```

The fields displayed for *LCP Status* are explained in *Negotiating the LCP Options* on page A-130. The fields displayed for *IPXCP Status* are explained in Table A-26.

Table A-26. Fields in (NCP) IPXCP Status Display

Field	Explanation
State	Shows the current and prior state of the IPXCP link. The states are:
Closed	The link has shut down via an administrative or peer request.
Request sent	The Annex has sent a configure request and is waiting for an answer.
ACK received	The Annex has received a configure ACK and is waiting for a configure request.
ACK sent	The Annex received and answered a configure request.
Open	IPXCP negotiation has completed successfully.
Closing	The link is in the process of closing. The Annex has sent a terminate request and is waiting for a terminate ACK.
Options	Shows the current values of the negotiated options. The *Local:* column displays the value suggested by the Annex. The *Remote:* column displays the value suggested by the remote client. The options are:
Network No	The 8-digit hexadecimal IPX network number of the remote client.
Node No	The 12-digit hexadecimal IPX node number of the remote client.
Compression	The kind of IPXCP header compression used. This can be *Telebit* or *None.*
Routing Proto	The routing protocol used by the Annex and (optionally) the client. This can be *RIP/SAP* or *None*.
Router Name	The name by which the Annex is known as an IPX router.

IPX Interfaces, Memory Buffers, Routes, and Servers

The CLI command **netstat –x** provides options for displaying information about:

- IPX in general.
- Using the **netstat -x** command itself.
- IPX network interfaces.
- The amount of memory available in the IPX buffer pools.
- IPX routes (RIPs).
- IPX servers.

The **netstat -x** syntax is:

netstat -x [**i** | **r** [*network_number*] | **s** [*server_name*] | **? | m**]
or
netstat -x [**i** | **r** [*network_number*] | **S** [*server_name*] | **? | m**]

IPX in General

Issuing the **netstat –x** command displays the number of NICs, RIPs, and of Service Advertising Protocol (SAP) services on the Annex. *NICs* indicates the number of active IPX interfaces (including **en0**) on the Annex, and *RIPs* indicates the number of Novell networks the Annex can reach.

The **netstat -x** command display looks like this:

```
annex: netstat –x
There are 2 NICs, 3 RIPs, and 4 SAPs
```

Using the netstat -x Command

Issuing the **netstat –x ?** command displays information about the use of **netstat –x**, as follows:

```
annex: netstat -x?
Usage: netstat -x
                -xi
                -xm
                -xr [network]
                -xs [server_name]
                -xS [server_name]
```

IPX Network Interfaces

Issuing the **netstat -xi** command displays information about the Annexes currently in use for dial-in or LAN-to-LAN routing. The following is a sample display.

```
Name   Network   Tics C0   NB   S0   Ipkts   Ierrs Opkts  0errs Collis

en0    00001234  2    n    y    n    21592   0     201380 0     0

asy18  00003456  4    n    y    n    72      0     98     0     0
```

The field headings in the above display indicate the following:

- *Name* is the interface name of the corresponding IPX port over which IPX dial-in or routing is currently occurring.
- *Network* is the number of the network to which interface *Name* connects.
- *Tics* indicate the amount of time associated with the cost of using interface *Name*. A tic is approximately 55 milliseconds.
- The *CO* field is not used.

- *NB* indicates whether or not this interface propagates NetBIOS information.
- *SO* indicates whether or not this interface propagates Server information only.
- *Ipkts* is the number of IPX packets received on this interface.
- *Ierrs* is the number of incoming IPX packets that contained errors.
- *Opkts* is the number of IPX packets transmitted over this interface.
- *Oerrs* is the number of outbound IPX packets that contained errors.
- *Collis* is the number of times a packet transmission was terminated due to a collision.

IPX Buffer Pools

Issuing the **netstat –xm** command displays the amount of memory available in the large and small IPX buffer pools. The Annex creates these buffer pools when it boots, allotting the appropriate amount of memory for the number of **auto_detect**, **auto_adapt**, and **ppp** ports configured. If you change these port modes, reboot the Annex so that it can allot the proper amount of buffer memory.

```
annex: netstat -xm

Large IPX Buffer Pool: Free = 0125 Total = 0125 Min =  0109

Small IPX Buffer Pool: Free = 0125 Total = 0125 Min =  0117
```

IPX Routes

Issuing the **netstat –xr** command displays the routes defined in the Annex’s IPX routing table. In the following example, **netstat –xr** displays five routes.

```
annex: netstat -xr

Network     Gateway          Tics     Hops     Interface

2d90ab99    0000a2816349     3        2        en0

00000042    0000a2816349     24       5        en0

00000043    0000a2816349     3        2        en0

00000044    0020af07dec4     3        2        en0

00001234    ffffffffffff     0        0        en0
```

The field headings in the above display indicate the following:

- *Network* is the number of a destination Netware network.
- *Gateway* is the number of the next hop on the path to *Network*. A gateway of ffffffffffff indicates a directly-attached network.
- *Tics* indicate the amount of time required to reach *Network* when *Gateway* is the next hop. A tic is approximately 55 milliseconds.
- *Hops* are the number of routers that must be crossed to reach *Network.*
- *Interface* is the network interface using the route.

Issuing the **netstat –xr** command followed by a network number displays the Annex route for that network. The following example shows how to display the route for network 42 (you can omit the leading zeros when specifying the network number):

```
annex: netstat -xr 42

Network     Gateway            Tics      Hops      Interface

00000042    0000a2816349       24        5         en0
```

IPX Servers

Issuing the **netstat -xs** command displays server names, types, and addresses.

```
annex: netstat -xs

OSCAR                     File Server
[2e80703c]000000000001          [0451]
CTEST                     Annex NAS
[00000055]00802d01d252[e480]
VENUS                     File Server
[00006501]000000000001[0451]
SMTPQ                     Advert Print
[00000043]000000000001[8060]
SNOWY                     Annex NAS
[00000063]00802d01ea57[e480]
```

From left to right, the fields in the previous displays are as follows.

- The first field is the server name, e.g., *`SUPT_TJB_INT.`* If the name is longer than 34 characters, **netstat –xs** displays only the first 34 characters.
- The second field is the server type, which can be:
  - *File Server*
  - *Job*
  - *Print*
  - *Archive*
  - *Job Queue*

- *NAS SNA Gate(way)*
- *TimeSync VAP*
- *Dynamic SAP*
- Annex NCS
- *Annex NAS*
- *Advert(ised) Print*
- *Btrieve (5.0) VAP*
- *SQL VAP*
- *TES-NetW(are) VMS*
- *NetW(are)* Access
- Named Pipes
- *NetW(are)* UNIX
- Netware 386
- *NETW(are) manage(ment) (type 0x6601)*
- *NETW(are) manage(ment) (type 0x6a02)*
- *Unknown <type>*

In the list above, text in parentheses is provided for clarity; **netstat –xs** does not display it.

- The third field is the server’s hexadecimal address, displayed in the format [*network*] *address* [*socket*].

Issuing the **netstat –xS** command displays an additional line of information for each server. The additional line contains the Annex route for the server.

```
annex: netstat -xS
HOBBESAnnex NAS      [00000012] 00802d009930 [e480]
  Gateway = [00000009] 0000a2816349  Hops = 2   IF = en0
ARAMIS NetWare 386   [0beef123] 000000000001 [8104]
  Gateway = [00000009] 0000a2816349  Hops = 3   IF = en0
ARAMIS File          [0beef123] 000000000001 [0451]
  Gateway = [00000009] 0000a2816349  Hops = 2   IF = en0
ROSA NetWare 386     [1a2a3b4c] 000000000001 [8104]
  Gateway = [00000009] 0000a2816349  Hops = 3   IF = en0
ROSA File            [1a2a3b4c] 000000000001 [0451]
  Gateway = [00000009] 0000a2816349  Hops = 2   IF = en0
```

When issued with a *server_name* argument after the **–s** or **–S** option, **netstat –sx** or **netstat –Sx** displays information for that specified server only.

Server names are typically in upper case.

IPX Frame Type and Network Number

Issued with no arguments, the CLI **stats** command displays various Annex statistics, including the IPX frame type of the Ethernet port and the Annex Netware network number.

The following is a sample **stats** display; IPX information is highlighted:

```
annex: stats

S/W Version: Remote Access Rx.x Build #2: Thu Jan 9 20:37:27 EDT 1995
H/W: Remote Annex 4000          H/W Rev: 36. ROM Rev 0811.
Comm: eth-aui&twi/64asy/1par    Mem: 5mDRM/64kEEPRM/16kSL1/16kSL2
Boot from: 132.245.88.5         Date: Thu Jan 9 13:27:50 1995 EDT
Image: oper.46.enet             Uptime: 15 hours 48 mins
Inet addr: 132.245.88.170       Subnet mask: 255.255.255.0
Ethernet addr:00-80-2d-00-b4-42 Broadcast addr: 132.245.88.255
Default domain: <unknown>
IPX: Frame_Type raw802_3/Network_Number 1234
```

(continued on next page)

```
Loading:
     CPU current/average = 1%/0%  procs active/max/limit = 87/88/800
     rescheds = 0/32  switches = 48/109401  activates = 49/109722
Mbufs:
     total=5400  free=3273  minimum free=3200  denied=0
Serial Ports:
     Total bytes:  rcv'd=24982  xmt'd=5934
     Errors:  parity=0  framing=0  fifo overruns=0
Parallel Ports:
     Total bytes:  xmt'd=0
Memory:
     total=5242880 avail=3894424 free=2073480 min free=1782488
     fails=0
```

If a 0 is displayed for the IPX network number, either the **ipx_frame_type** parameter was not configured properly or there is no IPX file server on the network.

IPX State

Issued with the –**o** option, the CLI **stats** command shows whether or not IPX is enabled (see *Enabling IPXCP* on page 11-274).

IPX Connections

The CLI **who** command displays specific information about an IPX connection, including what protocol the connection is using, the user name associated with the connection, where the connection is located, when the connection was created, how long the connection has been idle, and the address from which the connection was made. The following is an example:

```
annex: who

Port   What  User   Location     When     Idle Address
v1     CLI   ---    ---          10:00am       132.245.9.4
2      PPP   ---    ---          11:00am :20   [local]
```

Statistics for All Interfaces and for 802.2

Use the CLI command **netstat –i** to display statistics for current Annex interfaces and for the 802.2 data-link layer. An example follows:

```
annex: netstat –i
```

```
Name   Mtu   Network      Address    Ipkts  Ierrs  Opkts  Oerrs  Collis
en0    1500  132.245.66.0 worm       26563  0      15085  744    0
en0    1500  10000–20000  18062.79   1626   0      823    0      0
lo0    1536  127          127.0.0.1  0      0      0      0      0
asy2   604   18358        18062.79   0      0      0      0      0
asy16  1006  132.245.6    annex01    14770  0      7468   0      0
asy3   1500  192.9.200    zipwad     3453   0      3002   0      0

                   *** Hardware Interface Statistics ***

Ethernet Address:      00-80-2d-00-00-9b
Frames Received:       39861            Frames Transmitted:   45239
Bytes Received:        33965470         Bytes Transmitted:    29453
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
```

Chapter 12 AppleTalk

The Annex implementation of AppleTalk provides dial-in connectivity in a multi-protocol network. Using the Annex as a dial-in AppleTalk Remote Access (ARA) server, a remote ARA user can dial into the Annex and become a directly connected ARA network user. The Annex is transparent to the ARA user; it behaves like an AppleTalk end node.

AppleTalk Remote Access Protocol (ARAP)

ARAP allows Apple PowerBook and Macintosh computers to communicate with one another or with an AppleTalk network over standard telephone lines. A remote ARA user can dial into an AppleTalk network and take advantage of all the services available on the network, including:

- File transfer.
- Electronic mail.
- Database access.
- Printing.
- Mounting remote disks.

AppleTalk on the Annex supports ARAP V1 and V2.

Configuring the Annex for AppleTalk

Initially, all AppleTalk functions in the Annex are disabled. To enable the AppleTalk functions, the network administrator must obtain and enter the correct **option_key** parameter value and then reboot the Annex. The way to obtain a key depends on the configuration and type of Annex you purchased. Some option key values are physically attached to the bottom of the Annex. If the number is there, use it. If not, contact your supplier for an **option_key** value.

The **option_key** parameter enables a variety of Annex features, including **tn3270**, AppleTalk, and IPX. When requesting an AppleTalk **option_key** value from your supplier, be sure to mention any of the other **option_key** features currently enabled for your Annex.

After the reboot, the Annex automatically determines the appropriate network information, e.g., its AppleTalk node ID, etc. The AppleTalk-specific Annex parameters **a_router**, **zone**, and **node_id** are hints for the Annex to use at start up (*AppleTalk-specific Annex Parameters* on page 12-304 describes these parameters).

The Annex behaves like an AppleTalk phase II end node. At start-up, it listens for an AppleTalk router in the start-up network range and begins the process of finding its address. The Annex selects as its A_Router the first router it detects broadcasting an RTMP Route Data Request. It acquires an address within the A_Router's net-range and then uses a ZIP GetNetInfo and ZIP GetZoneList to find the network's zones; otherwise the Annex obtains an available address within the start-up range.

The Annex also installs a net-range route and an AppleTalk default route from the A_Router. If another router broadcasts an RTMP message, and its Ethernet address matches the address defined in the Annex parameter **a_router**, the Annex discards the current router information and tracks to this new router. If the Annex does not hear from the current A_Router for 50 seconds it selects a new A_Router. This 50 second hold-down prevents the Annex from bouncing between routers.

When ARA clients connect to an Annex port, the **node_id** parameter acts as a hint to acquire an address for the client. The Annex then installs a proxy *aarp* entry and the client's zone multicast address.

AppleTalk-specific Configuration Parameters

You can use either the host-based **na** utility, the local CLI **admin** command, SNMP, or the Annex Manager to configure Annex parameters.

The AppleTalk-specific configuration parameters are divided into two groups:

- AppleTalk-specific Annex parameters.
- AppleTalk-specific serial line port parameters.

For more details on using **na**, see *na Commands* on page C-1.
For more details on using the CLI commands, see *Using the CLI Commands* on page C-121.
For more details on using SNMP, see *Simple Network Management Protocol (SNMP)* on page B-41.
For more details on using the Annex configuration parameters, see *Configuration Parameters* on page C-33.
For more details on using Annex Manager, see the *Annex Manager User Guide*.

AppleTalk-specific Annex Parameters

The AppleTalk-specific Annex parameters are visible only when the **option_key** parameter contains the correct key value for the Annex. These parameters provide some AppleTalk protocol control, limits, and identification. Table A-27 lists these parameters; the following subsections describe them in detail.

If the **option_key** parameter is invalid, the Annex automatically disables AppleTalk.

Table A-27. AppleTalk-specific Annex Parameters

Parameter	Default	Description
a_router	00-00-00-00-00-00	The Ethernet address of the network’s A_Router.
default_zone_ list	""	This zone list is sent to ARAP clients as the local back-up to ACP.
node_id	0.0	The address the Annex tries to acquire at start-up.
option_key	""	Enables/disables AppleTalk.
zone	""	The AppleTalk zone for use at start-up.

Since AppleTalk uses dynamic addressing, AppleTalk addresses are acquired at boot time. The **a_router**, **zone**, and **node_id** parameters are hints for the Annex to use at start-up. If another AppleTalk node is using an address defined as a hint, the Annex chooses a different address. The **stats** command displays the run-time values for these parameters.

a_router

The Ethernet address of the network's A_Router. The Annex uses this value as a hint at start-up. When a Routing Table Maintenance Protocol (RTMP) message arrives from this Ethernet address, the Annex gleans the AppleTalk DDP address from the packet and tries to talk to the AppleTalk router. The address is a hexadecimal Ethernet address, e.g., 00-7F-12-33-44-55. The default is **00-00-00-00-00-00**.

default_zone_list

This zone list is sent to ARA clients as the local back-up to ACP. The parameter is a 100-character string with spaces separating the zones, e.g., **marketing engineering sales**. When this parameter is not set, the Annex provides the network zone list. The default is a **null string** ("").

You must use the backslash (\) character to escape embedded spaces.

node_id

This is the address the Annex tries to acquire at start-up. If this address is in use, the Annex must acquire a new node ID. The **node_id** is an AppleTalk address in the form *net.node*. Valid *net* values are **0** to **65534**; valid *node* values are **0** to **254**. The default is **0.0**.

option_key

The **option_key** parameter enables the AppleTalk-specific Annex parameters as well as the ARA protocol. AppleTalk commands, parameters, and port functions are enabled only after the correct key is set; after setting the key, the administrator *must* reboot the Annex. Each Annex requires a unique key value. The way to obtain a key depends on the configuration and type of Annex you purchased. Some option key values are physically attached to the bottom of the Annex. If the number is there, use it. If not, contact your supplier to obtain an **option_key** value.

The **option_key** parameter enables a variety of Annex features, including tn3270, AppleTalk, and IPX. When requesting an AppleTalk **option_key** value from your supplier, be sure to mention any of the other **option_key** features currently enabled for your Annex.

zone

The **zone** parameter provides the AppleTalk zone for use at start-up. It is a 32-byte string variable.This is the zone in which ARA clients are located unless overridden by security. The default is a **null string** ("").

AppleTalk-specific Serial Line Port Parameters

The **set port** command modifies the AppleTalk-specific serial line port parameters; the **show port** command displays them. Table A-28 lists these per-port parameters; the following subsections describe them in detail.

Although the **mode** parameter is not AppleTalk-specific, it does affect AppleTalk and therefore is included in this section.

Table A-28. Per-port AppleTalk Parameters

Parameter	Default	Purpose
arap_v42bis	Y	Enables/disables V.42bis compression during an ARA session.
at_guest	N	Allows ARA guest login service.
at_nodeid	0.0	The node ID given to an ARA client during connection establishment.
at_security	N	Enables/disables ACP service for this port.
mode	cli	Sets the mode for access to a serial line port.

arap_v42bis

The **arap_v42bis** parameter enables/disables V.42bis compression during an ARA session. A **Y** enables the parameter; an **N** disables it. The default is **Y**.

If you disable this parameter, you may want to change the Communications Control Language (CCL) script for the remote modem to improve performance. Sample CCL scripts are included in the software distribution.

at_guest

The **at_guest** parameter allows guests to log into an ARA service. When this parameter is enabled and a client requests guest access, the Annex asks ACP for user name (guest) privileges. A **Y** enables guest privileges; an **N** disables guest privileges. The default is **N**.

at_nodeid

The **at_nodeid** parameter defines the node ID hint used for an ARA client during connection establishment. This parameter value is an AppleTalk address in the form *net.node*. The valid *net* values are **0** to **65534**. The valid *node* values are **0** to **254**. The default is **0.0**.

at_security

The **at_security** parameter enables/disables ACP service for this port. When both **at_security** and **enable_security** are set, the Annex uses ACP to get per-user security information about the client (authentication, logging, and zone access) from the **acp_userinfo** file (see *Creating the acp_userinfo File* on page A-454). If **at_security** is not set, the Annex uses only local security (**port_password** and **username** for authentication, and the **default_zone_list**). A **Y** enables this parameter; an **N** disables it. The default is **N**.

mode

The **mode** parameter sets the type of access for a serial line port; it determines whether access is initiated by a device to the Annex or from the network through the Annex to the device.

Setting **mode** to **arap** provides a port that can perform as a network interface using ARA; the default mode for a port is **cli**.

Setting **mode** to **auto_detect** provides a port that automatically determines the protocol of an incoming packet and converts to **arap**, **ipx**, **ppp**, **slip**, or **cli** mode accordingly.

A port set to **auto_adapt** mode automatically detects whether packets are incoming or outgoing. For incoming packets, the port behaves as if it were set to **auto_detect** mode, then determines the incoming protocol and converts to the appropriate mode, as described. For outgoing packets, the port operates in **slave** mode (see *Slave Ports* on page A-65).

CLI AppleTalk Commands

The Command Line Interface (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. Table A-29 lists the CLI commands for use with AppleTalk; the following subsections describe them (*Using the CLI Commands* on page C-121 describes all CLI commands).

Table A-29. CLI AppleTalk Commands

Command	Description
arap	Converts a CLI line into an ARA connection. When the port is reset, it reverts to its original mode. After entering the command, the Annex prompts: *Annex switching line to ARAP*. The **arap** command does not apply to VCLI connections.
arp	This superuser command displays ARP cache on the Annex.
netstat	Displays information about AppleTalk interfaces.
ping	This superuser command generates AppleTalk Echo Packets (AEP).
stats	Displays AppleTalk information.
who	Displays a line’s type as ARA.

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.

arap

The **arap** command converts a CLI line into an ARA connection. Resetting the port returns the CLI to its original mode. The syntax is:

arap

The command display looks like this:

```
annex: arap
Annex switching line to ARAP.
```

arp

The **arp** command displays and, optionally, modifies the IP-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. Table A-30 lists the arguments for this command.

Although the **arp** command shows AppleTalk information, you cannot manipulate it. Since **arp** interprets all address as IP addresses, if you try to delete an AppleTalk address such as 1.123 using **arp –d**, the ARP table entry 1.0.0.123 is deleted.

The syntax is:

arp [**–ads**] [*host*] [*addr*] [**temp** | **pub**]

Using either the *host* or the **–a** argument, **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:

```
annex01# arp –a

xenna (192.9.200.95) at 08-00-4C-00-2a-c0 tt1=20
2356.189 at 08-00-4e-34-22-39 tt1=16
```

Table A-30. Arguments for the arp Command

Argument	Description
host	Displays the current ARP table entry for that host.
addr	Displays the current ARP table entry for that address.
–a	Displays all entries in the table.
–d	Deletes the entry specified with *host*.
–s	Creates an entry for the host, specified using either *host* or an Internet address, at the hardware address specified using *addr*. If you do not include **temp** or **pub**, 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.

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. Table A-31 describes the arguments for **netstat** that relate to AppleTalk. The syntax is:

netstat [**–AaCfgimnpQRrSstx**[**–i** | **–r** [*network_number*] | **–sS** [*server_name*] | **? | –m**] **z**] *port*

The display format varies according to the options selected and the network protocols implemented for the Annex.

See *netstat* on page C-158 for details on the **netstat** command. See *Displaying Network Statistics* on page B-1 for sample display formats.

Table A-31. AppleTalk-related Arguments for the netstat Command

Argument	Description
–i	Displays interface statistics. AppleTalk addresses display as *net.node* in decimal, where *net* is 16 bits and *node* is 8 bits. ARA interfaces display as *asy* plus the unit number.
–ia *port*	Displays statistics for a specific Annex ARAP interface.
–ra	Displays only AppleTalk routes.
–s	Displays network protocol statistics. LAT statistics display only if the correct **lat_key** value is set. AppleTalk statistics display only if the correct **option_key** value is set.
–z	Displays the network zone list.

Addresses display as either *host***.***port* or *network***.***port*. The latter displays if a socket's address does not include a specific host address. Known host names are displayed; otherwise, the Internet addresses are displayed. Unspecified or wildcard addresses and ports appear as an asterisk (*).

The **netstat –i** command display looks like this:

```
annex01# netstat -i

Name Mtu  Network      Address   IpktsIerrsOpkts OerrsCollis
en0  1500 132.245.66.0worm      265630     15085 744   0
lo0  1536 127         127.0.0.10    0     0     0     0
asy2  604 18358       18062.79 0    0     0     0     0
```

(continued on next page)

```
            *** Hardware Interface Statistics ***

Ethernet Address:        00-80-2d-00-42-30

Frames Received:        1930      Frames Transmitted:   693

Bytes Received:         192956    Bytes Transmitted:  48501

CRC Errors:             0         Alignment Errors:       0

Bad Type/Length Fields: 1          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:    5         Max Collision Retries:  0

        *** IEEE 802.2 Data Link Layer Statistics ***

802.2 packets received: 1        802.2 packets sent:      0

ATALK packets sent:     0        AARP packets sent:       0

ATALK packets received: 0        AARP packets received:   0

Unknown 802.2 types:    0        Unknown 802.2 SAP's:     0

Unknown SNAP org codes: 0        Unknown SNAP ether types:0
```

The **netstat –ia 2** command display looks like this:

```
annex01# netstat –ia 2

*** ARAP Statistics ***
ATALK sent:          52    ATALK sent dropped:   0
ATALK received:      0     ATALK received dropped:0
ARAP Version 1
MNP received:        17683 MNP fcs errors:       0
MNP internal errors: 0     MNP sent:             3590
MNP retransmit:      0
```

The **netstat –z** command display looks like this:

```
annex01# netstat -z

eng lab        test lab         project B     account
techpubs       sqa              west sales    HR
market1        marcom           east sales    project C
market2        project A        central sales
```

ping

The CLI superuser **ping** command elicits an ICMP Echo Response from a specified host. The command prints a line of output for each response returned. The syntax is:

ping [**–artv**] *host* [*data_bytes* [*count*]]

Table A-32 explains the **–a** argument, which is the only one related to AppleTalk.

Table A-32. AppleTalk-related Argument for the Superuser ping Command

Argument	Description
–a	Generates AppleTalk Echo Protocol (AEP) echo request packets to a target node. Displays the time the packet took to turn around.

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. For more details on the **ping** command, see *ping* on page C-161.

The **ping** display for an AppleTalk node looks like this:

```
annex01# ping –a 10557.31
PING 10557.31. 94 data bytes
94 bytes from 10557.31: aep_seq=0. time=7. ms
94 bytes from 10557.31: aep_seq=1. time=6. ms
94 bytes from 10557.31: aep_seq=2. time=5. ms
94 bytes from 10557.31: aep_seq=3. time=5. ms
```

stats

The **stats** command displays Annex statistics. The AppleTalk statistics follow the *Default domain* field; Table A-33 describes the AppleTalk fields in the **stats** display (these fields appear in bold type in the following sample display). For more details on the **stats** command, see *stats* on page C-184.

The **stats** command display looks like this:

```
annex: stats

S/W Version: Remote Access Rx.x Build #2: Thu Sep 14 20:37:27 EDT 1995
H/W: Remote Annex 4000          H/W Rev: 36. ROM Rev 0811.
Comm: eth-aui&twi/64asy/1par    Mem: 5mDRM/64kEEPRM/16kSL1/16kSL2
Boot from: 132.245.88.5         Date: Thu Sep 21 13:27:50 1995 EDT
Image: oper.46.enet             Uptime: 15 hours 48 mins
Inet addr: 132.245.88.170       Subnet mask: 255.255.255.0
Ethernet addr:00-80-2d-00-b4-42 Broadcast addr: 132.245.88.255
Default domain: xylogics.com
Apple:Node 20801.233 A_Router 20844.132 Zone XyloEng
Loading:
     CPU current/average = 1%/0%  procs active/max/limit = 87/88/800
     rescheds = 0/32  switches = 48/109401  activates = 49/109722
Mbufs:
     total=5400  free=3273  minimum free=3200  denied=0
Serial Ports:
     Total bytes:  rcv'd=24982  xmt'd=5934
     Errors:  parity=0  framing=0  fifo overruns=0
Parallel Ports:
     Total bytes:  xmt'd=0
Memory:
     total=5242880 avail=3894424 free=2073480 min free=1782488
     fails=0
```

Table A-33. AppleTalk Fields in the stats Display

Field	Description
Node	The AppleTalk address acquired by the Annex.
A_Router	The AppleTalk address of the current A_Router.
Zone	The zone in which the Annex is located.

who

The **who** command displays information about current users on the Annex’s ports. This command also displays current users on other Annexes, and on remote hosts, if those hosts have **fingerd** running for **who** @host. The command accepts one or more arguments (see *who* on page C-231 for more details).The syntax is:

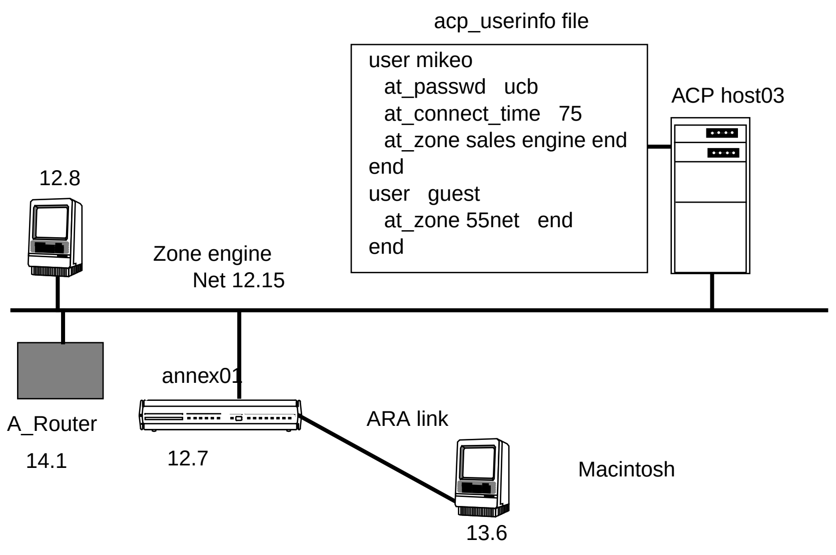
who [[**h**=]*host* | [**u**=]*user* |[**p**=]*port* | *@host* |*user@host* | **–l** *@host*]

AppleTalk over ARA

AppleTalk over ARA allows Apple PowerBook and Macintosh computers to communicate with one another or with an AppleTalk network over standard telephone lines. An ARA user can dial into a remote AppleTalk network and use all the available services as if that user is physically connected to the network through EtherTalk.

Setting Port Parameters for AppleTalk over ARA

Figure A-28 illustrates the following sample settings. In this configuration, a Macintosh connected to an Annex through an ARA link appears to the network as an attached node.



```
user mikeo
  at_passwd   ucb
  at_connect_time   75
  at_zone sales engine end
end
user   guest
  at_zone 55net   end
end
```

Figure A-28. Connecting a Macintosh Using ARA

Setting up a port for AppleTalk over ARA requires that you configure the port for both an inbound modem and an ARA link (see *Setting Remote Annex Port Configuration Parameters for Modems* on page A-101 for more details). Using a central security server for dial-up access allows users to have their own addresses.

- Set the **mode** parameter to **auto_adapt** or **auto_detect**.
- Setting the **type** parameter to **dial_in** registers the user with the **who** database as soon as the ARA process attaches to the line.
- Enable ARA security by setting **at_security** to **Y**. ACP and port access is logged in the the ACP log file.
- Use the supplied defaults for the **data_bits** (8), **stop_bits** (1), and **parity** (none) parameters.

ARA is an 8-bit protocol. If **data_bits** is set to **7**, and **parity** is not set to **none**, the Annex forces the **data_bits** setting to **8** and the **parity** setting to **none**. Otherwise, the Annex generates an error message for the port.

- To configure the port for EIA flow control, set the **control_lines** parameter to **both** and the **input_flow_control** and the **output_flow_control** parameters to **eia**.
- Set the **arap_v42bis** parameter to **Y**.
- Setting the **at_guest** parameter to **Y** enables guest access.
- The **at_nodeid** parameter defines the AppleTalk node address for the remote Apple PowerBook or Macintosh.

ARA Security

The Annex provides comprehensive security features that assist you in securing your Annexes and the network from unauthorized access. Using 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.

If you are using host-based security, you must define the user name and password in the **acp_userinfo** file (see *Creating the acp_userinfo File* on page A-454).

The Annex provides protection through the use of an administrative password that controls access to the superuser CLI commands. This password can also protect access to an Annex through **na**. The security system provides audit trails that monitor users and their activities. The Annex also provides the source code for the Access Control Protocol (ACP) security system, and the flexibility to integrate Annex security with existing security for a network-wide system.

The following subsections briefly describe Annex security as it relates to ARA. For a detailed description of ACP, host-based security, and the **acp_userinfo** file, see *Using Remote Annex Security* on page A-421.

Security Features

The Annex implementation of ARA provides three areas of security:

- ARA security.
- Zone security.
- Logging.

ARA Security

The basic ARA security features are:

- **username and password authentication**

  The Annex authenticates the client using Apple's DES encryption algorithm. To define a user name and password for a registered (as opposed to guest) user, see *Creating the acp_userinfo File* on page A-454.
- **guest access**

  The Annex allows anonymous access to the network. Restrictions can be applied to *guests* by setting up an ACP *guest* profile with limitations. For more details, see *at_zone* on page A-466.
- **connection timer**

  The connection timer is stored in the **acp_userinfo** file. For more details, see *Creating the acp_userinfo File* on page A-454.

Zone Security

Every user can have a zone list assigned via remote ACP. If a list is not available via ACP, the Annex provides all the zones it has learned from the network. If local security is used, use the per Annex parameter **default_zone_list.** For more details, see *at_zone* on page A-466.

Logging

The Annex logs activity and errors from the ARA session. The log is accessed via remote ACP and **syslog** (see *Logging User and Annex Events* on page B-26 for more details).

Network-Visible Entity (NVE) Filtering

NVE filtering controls a remote access Apple user's view of network resources: when using *Chooser* to select resources, only the resource set defined for the user by the administrator will be visible. The administrator can specify the NVE filter on a per-user basis. This feature complements the exisiting zone list, described above, by offering a higher level of control.

The administrator uses the **nve_filter** entry in the **acp_userinfo** file to specify a list of filters on a per-user basis. Please see *at_nve_filter* on page A-468 for detailed information on creating **nve_filter** entries.

This method of limiting NBP traffic is not secure, and can be circumvented by a person willing to write some code to probe the network without using NBP. Also, this feature has no local Annex security equivalent.

AppleTalk over PPP

AppleTalk over PPP allows Apple PowerBook and Macintosh computers to connect as an endpoint node to an AppleTalk network. The same Macintosh can also simultaneously run IP over the connection, allowing the user to use either IP or AppleTalk services as the need arises.

When the Annex opens a PPP connection, it negotiates for link-level options, and then runs an optional security phase to authenticate the user. Finally, the two ends negotiate for network control protocol (NCP) options. The link is then opened and becomes a generic interface for the Annex. An AppleTalk point-to-point link is configured, enabled, and disabled using AppleTalk Control
Protocol (ATCP).

The Annex implementation of ATCP currently supports dial-in only.

How to use the CCL Converter

The Macintosh CCL Converter application converts the CCL modem configuration file to allow access to the Annex via ARAP (Versions 1 and 2). Typically, the CCL file sets up the modem and issues the dial command. When you dial into the Annex from a Macintosh, the Macintosh CCL modem configuration file also controls the connection until protocol negotiation is complete.

The CCL Converter supports SecureID, Enigma, and CLI Dialback.

Configuration

The Annex administrator can configure the CCL Converter Application to connect to an Annex port mode that is set to **arap**, **auto_adapt**, **auto_detect**, or to **cli**. Connecting to a CLI port is useful only for networks requiring port passwords or SecurID security. The Annex host-based (ACP) security is available via **arap** and **auto_detect** ports.

As administrator, configure the CCL Converter on the Macintosh as follows:

1. **Using a Macintosh-based** ftp **program, such as Fetch, that is set to MacBinary, copy the CCL Converter from the following directory on your UNIX load host:**

   ```
   usr/annex/src/examples/ccl_scripts
   ```

   The file name is CCL Converter.

   You must ‘escape out’ the space between the word *CCL* and the word *Converter* in the CCL Converter file name. Do this by entering the file name inside a pair of double quotes, e.g., “CCL Converter”.

2. **From the Macintosh Settings menu, select ARAP V1/Autodetect Delay, CLI Security, or both, depending on the type of Annex security you desire, as shown in the following table.**

Table A-34. Selecting Security Type

CCL Setting	Port Parameter Settings	Security Result
ARAP V1/Autodetect Delay	**mode**: **arap** or **auto_detect** **at_security**=**Y**	Annex ARAP ACP security.
CLI Security	**mode**: **cli** or **auto_detect** **at_security**=**N**	Normal (non-ARAP) ACP security, including port password and SecurID, if configured.
ARAP V1/Autodetect Delay *and* CLI Security	**mode**: **cli** or **auto_detect** **at_security**=**N**	Normal (non-ARAP) ACP security, including port password and SecurID, if configured.
CLI Security	**mode**: **cli** or **auto_detect** **at_security**=**Y**	Both ARAP ACP security and normal (non-ARAP) security.
ARAP V1/Autodetect Delay *and* CLI Security	**mode**: **cli** or **auto_detect** **at_security**=**Y**	Both ARAP ACP security and normal (non-ARAP) security.
CLI Security	**mode**: **arap**	Connection fails.
ARAP V1/Autodetect Delay *and* CLI Security	**mode**: **arap**	Connection fails.

3. **If, in Step 2, you selected V1/Autodetect Delay, configuration is complete. Skip to *Running the Application*, below.**

   If you selected either CLI security or both, and you have modified the CLI prompt and/or the ACP Policy file on the Annex, you must use the CCL Converter's Customize menu to configure for the Macintosh any prompts you changed on the Annex.

   For example, Annex administrators typically change the CLI prompt - suppose your customized Annex prompt is the Annex name followed by the port number. Under Customize on the Macintosh, select the *CLI Prompt...* menu item. A dialog box now appears. In the dialog box, enter the Annex name portion of the Annex prompt, but not the port number (since the user will be connecting to different ports) and either click **OK** or press **Return**. Make sure the Annex name you enter is enclosed in quotes.

   Once all modifications are made, the configuration is complete. Go to *Running the Application*, below.

Running the Application

1. **Double-click on the CCL Converter application.**
2. **From the File menu, select Open (to convert a single file) or Open Folder (to convert a whole directory of files).**

   When you select a file, the application creates a new file containing the conversion, and appends ANNEX to the filename. For example if the selected file is named AE Datalink PB, the application creates a new file named AE Datalink PB ANNEX. Names longer than 31 characters are truncated.

   Files with exactly 31 character names cannot be converted and cause the application to abort.Quit the CCL Converter.

3. **Select the converted CCL file from the remote access client (see your Apple Remote Access Client® documentation).**

Chapter 13 Printers

The Annex supports the standard **lpr**/**lpd** printing protocol along with two proprietary utilities (**aprint** and **rtelnet**) that enable hosts to send print requests to a remote printer.

- The **lpr/lpd** protocol is a standard system utility for printing which is typically distributed with systems that support TCP/IP.
- The **aprint** utility provides a unidirectional interface; it can send print jobs to both the serial and parallel ports. Printers using **aprint** must be output only devices.
- The reverse **telnet** daemon, **rtelnet** is a bidirectional interface that can send print jobs to serial, parallel, and PostScript-compatible printers.

The **aprint** and **rtelnet** utilities do not provide spooling capabilities, but you can use them as transparent utilities for other printer spooling systems, e.g., **lpd** (BSD) or **lp** (System V). For more details on using these utilities, see *Utilities* on page C-235.

Remote Annex Server Tools for Windows NT® does not support the **aprint** or **rtelnet** utilities.

Printer Cables

When using EIA flow control (also known as hardware handshake, hardware flow control, and RTS/CTS flow control), some printers may require a special cable. This cable should have the transmit data and ground leads connected from the printer to the Annex port as normal. The flow control signal generated by the printer (the location is printer-specific) should be connected to the CTS signal on the Annex: this is pin 4 on a terminal cable and pin 5 on a modem cable. See the appropriate Annex Hardware Installation Guide for serial cable wiring diagrams.

Configuring Parameters for a Serial Printer

Consider the following port configuration parameters when attaching a printer to a serial port (see *Configuring Ports* on page A-49 for more details):

- 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.
- Set the **allow_broadcast** parameter to **N**.
- For EIA flow control, set the **control_lines** parameter to **flow_control** and the **input_flow_control** and **output_flow_control** parameters to **eia**.
- For XON/XOFF set the **control_lines** parameter to **none** and the **input_flow_control** and **output_flow_control** parameters to **start/stop**.

Printers that use flow control with both EIA signals and XON/XOFF (^S and ^Q characters) should have the attached port configured for XON/XOFF, not for EIA (i.e., set **input_flow_control** and **output_flow_control** to **start/stop**).

If the port is configured for EIA signals, the Annex uses flow control via EIA. However, the Annex will send the XOFF character to the host; typically, this is not the desired result.

Configuring Parameters for a Parallel Printer

Use the **show printer** command to display the following parallel printer port parameters: **hardware_tabs**, **map_to_upper**, **printer_width**, **type**, **printer_crlf**, and **tcp_keepalive**. Use the **set printer** command to change these parameter settings.

When printing binary data, you must set the following parameters to these values:

• **map_to_upper**	**N**
• **printer_width**	**0**
• **hardware_tabs**	**Y**
• **printer_crlf**	**N**

The **type** parameter defines the printer interface. The options are **dataproducts** and **centronics**; the default is **centronics**.

There are two **type** parameters: one for the serial ports and one for the parallel printer port(s).

Printing from a Host using lpr/lpd

Using the **lpr/lpd** protocol, you can configure your host to treat the Annex as a remote printer; you can also designate the Annex port or rotary through which the data should transmit to the printer.

Solaris 2.4 compatibility note: The **lpr** and **lpq** utilities on this host are integrated into the Solaris printing daemon and do not function in exactly the same manner as the BSD-type utlities of the same name. In particular, the Solaris **lpq** utility shows a locally cached state, and does not show the normal list of ports in use (which the Annex produces in response to a query). Keep this in mind when testing these utilities.

Editing the /etc/printcap File for a BSD Host

1. **Set up the Annex as a standard remote printer.**
2. **Create an entry for the remote printer in** /etc/printcap**.**

   In the following sample entry, the remote printer is named *ptr* (which is the name of a rotary configured on the Annex named *jdc*). The local name for the printer on this host is defined as *myprt*.

```
myprt|Remote printer on jdc rotary ptr:\
     :lp=:rm=jdc:rp=ptr:mx#0:\
     :lf=/var/adm/jdc-errs:\
     :sd=/usr/spool/jdc:
```

   You can specify a serial port by entering the port number:

```
myprt|Remote printer on jdc \
     :lp=:rm=jdc:rp=1:mx#0:\
     :lf=/var/adm/jdc-errs:\
     :sd=/usr/spool/jdc:
```

   You can specify a parallel port as as *lp* (or *lp#* for units with multiple parallel ports, where # is 1–9):

```
myprt|Remote printer on jdc \
     :lp=:rm=jdc:rp=lp:mx#0:\
     :lf=/var/adm/jdc-errs:\
     :sd=/usr/spool/jdc:
```

Installing on a System V Host

1. **Set up the Annex as a standard remote printer.**
2. **Create an entry for the remote printer.**

   In the following sample entry, the remote printer is named *ptr* (which is the name of a rotary configured on the Annex named *jdc*). The local name for the printer on this host is defined as *myprt*.

```
lpadmin –p myprt –s ‘jdc!ptr’
```

You can specify a serial port by entering the port number:

```
lpadmin –p myprt –s ‘jdc!1’
```

You can specify a parallel port as as *lp* (or *lp#* for units with multiple parallel ports, where # is 1–9):

```
lpadmin –p myprt –s ‘jdc!lp’
```

Printing from a BSD Host using aprint or rtelnet

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 a direct command, or 2) as part of an output filter of a BSD spooling system.

Remote Annex Server Tools for Windows NT® does not support the **aprint** or **rtelnet** utilities.

Using aprint as a Direct Command

When using **aprint** as a direct command, specify the Annex and port using the **–A**, –**p**, and **–L** arguments (Table C-77 on page C-236 describes the arguments for **aprint**).

A banner page is not printed when using **aprint**. If you want a banner page, create a shell script that calls **aprint** and place it in a publicly executable area; users can execute the shell instead of calling **aprint**.

The following sample shell script, called **laser1**, takes the file(s) to be printed as an argument, adds a banner page, and calls **aprint** using the **–A** and **–L** arguments:

```
#!/bin/sh
for file in $*
do
     makebanner $file
     cat $file
done | /usr/annex/aprint -Aannex01 -L15
```

Users can issue the shell script **laser1** to print:

```
# laser1 file1 file2
```

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

Integrating aprint with the lpd Spooler

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 in the **/etc/printcap** file for the BSD **lpd** program that uses the conversions and expansions.

The Filter Program filt.c

The *C* program **filt.c** is included with the Annex software distribution in the **src/examples** directory. This program 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**.

The **filt.c** program is compatible with SunOS; other UNIX operating systems may require modifications to the program. Also, **filt.c** assumes that the Annex programs are installed in the directory **/usr/annex**. If this is not true, change the definition of APRINT.

The filters are defined by the FILTER definition at the top of the program.

The **filt.c** program is not compatible with Remote Annex Server Tools forWindows NT®.

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

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

On some systems, the programs *awk* and *expand* may reside in different directories than those shown here; if so, change this line accordingly.

After defining a filter, compile and link the **filt.c** program to a name that specifies the Annex port to which the output should go. For example, to send output to port 15 on the Annex called *annex01*, **filt.c** should be linked to *annex01.15*. The **filt.c** program looks like this:

```
#include <stdio.h>
#define SEPARATOR'.'
#define APRINT"/usr/annex/aprint"
/*
 * If you have 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;
     char                *basename;
     int                 length;
     basename = (char *)strrchr(argv[0],'/');
     if(basename == (char *)NULL) {
                         basename = argv[0];
     } else {
                         basename++;
     }
```

(continued on next page)

```
        p = (char *)strrchr(basename,SEPARATOR);
        if(p == (char *)NULL){
        fprintf(stderr,"Error:name not of form\
        annex%cport\n",SEPARATOR);
                        exit(1);

        }

        length = (int)(p-basename);
        strncpy(annex,basename,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);
}
```

Running a Shell Script Filter

If you are running on a BSD system, you can replace the **filt.c** program with a 2-line shell script. For example:

```
#!/bin/sh
sed 's/$/^M/' | /usr/annex/aprint -Aannex01 -L15
```

The line `#!/bin/sh` is critical because it allows the UNIX kernel to execute this shell as a Bourne shell script.

Editing the /etc/printcap File

If the filter program *annex01.15* is in the directory **/usr/annex**, create an entry in **/etc/printcap** that looks 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 using the **mknod** program:

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

In this example, the *3 2* entries are 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.

Test the printer by issuing the following **lpr** command:

```
# lpr -pannexprt /etc/group
```

Integrating rtelnet with the lpd Spooler

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. 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 using** rtelnet**. This example creates a device that allows a printer on port 16 of the Annex called *annex02*. The** –b **argument is included because the printer uses binary data which may be scrambled by Telnet's carriage return/line feed conventions.**

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

```
# rtelnet -br annex02 16 /dev/annexserialport
      annexprt|ap|serial printer on annex02:\
     :lp=/dev/annexserialport:\
     :sd=/usr/spool/annexserialport:\
     :lf=/usr/adm/lpd-errs:pw#80:fs
```

   For a parallel port, add an entry to the **/etc/printcap** file for the printer. This entry should appear as if the printer is connected directly to a port called
   /dev/annexparallelport:

```
# rtelnet -Pbr annex02 S901 /dev/annexparallelport
      annexprt|ap|parallel printer on annex02:\
     :lp=/dev/annexparallelport:\
     :sd=/usr/spool/annexparallelport:\
     :lf=/usr/adm/lpd-errs:pw#80:fs
```

3. **Test the printer:**

```
# lpr -pannexprt /etc/group
```

Printing from a System V Host using aprint or rtelnet

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 a direct command (see *Using aprint as a Direct Command* on page 13-331), and 2) as part of an output filter of a System V **lp** spooling system.

Remote Annex Server Tools forWindows NT® does not support the **aprint** or **rtelnet** utilities.

Integrating aprint into the lp Spooler

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*]

Table A-35. Arguments for the lpadmin Command

Argument	Description
–p*printer*	Specifies a new printer.
–v*device*	Associates a device with the printer; *device* is the path name of a file that is writable by **lp**.
–e*printer*	Copies the interface program of the specified printer to the new printer specified with **–p**.
–i*interface*	Defines the full path name for a new interface program.
–m*model*	Specifies a model interface program for the printer; *model* is a model interface supplied with the System V **lp** spooling system.

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 creates and uses an interface file with the path name **/usr/spool/lp/annex_printer** which was created specifically for a printer called *p_annex* (*Sample System V Interface File for aprint* on page 13-338 provides a sample file).

1. **Shut down the lp scheduler (this disables all printers):**

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

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

```
# lpadmin –pp_annex –v/dev/null –i/usr/spool/lp/\
  annex_printer
```

3. **Enable the printer:**

```
# enable p_annex
```

4. **Allow the queue to accept jobs:**

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

5. **Restart the lp scheduler:**

```
# /usr/lib/lpsched
```

6. **Test the printer:**

```
# lp –dp_annex /etc/group
```

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**.

In the following line of the sample interface file, the ^*M* is an embedded Control-M in the file:

```
| sed ‘s/$/^M’ | /usr/annex/aprint -Aannex08 -L4 -f
```

Additionally, you can replace the following part of the line:

```
| sed ‘s/$/^M’
```

to:

```
awk '{ printf("%s\r\n",$0) }'
```

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

#Change the next two lines to direct output to the correct
place.
#ANNEXLINE is port number to use, 0=parallel printer
ANNEX=annex01
ANNEXLINE=1

#Change this line if your Annex software is installed in a
#different directory
PATH=$PATH: /usr/annex
export PATH
#

TEMPFILE=/usr/spool/TMP.$$

# This will be executed when a request is cancelled
trap "echo ’\n\n\n\nRequest Cancelled’;\
  echo ’\-14\c’;\
  sleep 30;\
  exit 0" 15
#
#The following three lines are added for aprint...
while true #loop until aprint is successful
do
(
x=”XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX”
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: -f51‘
```

(continued on next page)

```
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 $i + 1‘
done
#
# The following 8 lines apply only to aprint
#
# ) | /usr/annex/aprint -Aannex08 -L4 -f
) | sed ’s/$/^M’ | /usr/annex/aprint -Aannex08 -L4 -f

   if test $? = 0   # check aprint return code
   then
      break   # good, all done..
   fi
   sleep 30   # bad, wait a while and try again..
done

exit 0
```

Integrating rtelnet with the lp Spooler

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_printer*.

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 (this disables all printers):**

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

3. **Create a character special device using** rtelnet**:**

   ```
   # rtelnet –Pbr annex01 s901 /dev/s_pdev
   ```

   This command creates the character special file **/dev/s_pdev** and creates an rtelnet connection between the associated pty device and port 12 on the Annex called *annex01*.

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

   ```
   # lpadmin –ps_printer –v/dev/s_pdev –mdqp10
   ```

5. **Enable the printer:**

   ```
   # enable s_printer
   ```

6. **Allow the queue to accept jobs:**

   ```
   # /usr/lib/accept s_printer
   ```

7. **Restart the lp scheduler.**

   ```
   # /usr/lib/lpsched
   ```

8. **Test the printer:**

   ```
   # lp –ds_printer /etc/group
   ```

Host lpq Command

The host **lpq** command provides the status of all ports in use by **aprint** or **lpd** (assuming that the named printer on the host is configured to point to an Annex port or rotary). The Annex generates one line of text for each active **aprint** or **lpr**. The text following those lines (starting with *calvin: sending to cody* in the following example) is generated by the host's **lpq** command and may vary depending on the type of host used.

```
% lpq –Pmyprt
No entries.
% lpq –Pmyprt
lpr job on serial port 4 from host 132.245.33.8.
aprint job on serial port 9 from host 132.245.33.71.
lpr job on serial port 17 from host 132.245.33.8.

calvin: sending to cody
Rank Owner             Job       Files     Total Size
1st  cobb              15        standard input1643 bytes
2nd  cobb              928       /etc/motd 65 bytes
```

In the previous example, the printer name, *myprt*, is the name assigned to the printer by the host system; this name is local to that system (in this case *calvin*) and is not related to the Annex configuration. Internally, in a description file residing on the host, is a mapping that indicates which Annex (in this case *cody*) and which port or rotary (in this case *17*) to use for jobs and requests directed to that printer.

Book A

Chapter 14 Configuring Hosts and Servers

This chapter describes configuration requirements for hosts providing the following Annex services:

- Accessing 4.2BSD hosts.
- Installing the ACE/Server software.
- Setting up the file server.
- Parsing the configuration file.
- Setting up the configuration file.
- Setting up the **motd** file.
- Setting up an Annex as a boot server.
- Self-booting without a local Ethernet interface.
- Installing a time server.
- Dump host services.
- Setting up name servers.
- Setting up a host for 4.3BSD syslogging.
- Configuring LAT services.

Accessing 4.2BSD Hosts

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

Add the Annex to the **/etc/hosts** file on each 4.2BSD host. Add the new entry near the beginning of the file because UNIX software searches this file sequentially.

Installing the ACE/Server Software

If you are using the Annex's SecurID feature, you must install the ACE/ Server software before installing the Annex software.

After compiling the ACE/Server installation, the file **sdconf.c** contains the network addresses for the ACE/Server host. See *Using the SecurID Card* on page A-524 and the *ACE/Server Manual* for more details.

Remote Annex Server Tools for Windows NT® does not support the ACE/Server.

Setting Up the File Server

The Annex can boot via the block file server (**bfs**) program using **erpcd** or via **tftp**.

Installing Software Using bfs

Setting up a file server for a **bfs** installation involves loading, compiling, and installing Annex source code on the host. This process has four stages (see the *Annex Software Installation Notes* for more details):

- Loading the software from the media into a directory.
- Running the **install-annex** script.
- Editing the **/etc/services** file.
- Starting **erpcd**.

Bfs booting is automatically enabled when you install Remote Annex Server Tools for Windows NT®. The script and file editing steps are not required.

When the installation is complete, by default, the image and configuration files are located in the directory **/usr/spool/erpcd/bfs**; the utilities are located in the directory specified during the **install-annex** sequence (the default is **/usr/annex**).

Installing Software Using the tftp Protocol

Setting up a file server for a **tftp** installation involves loading the Annex software onto the host. This process has three stages (see the *Annex Software Installation Notes* for more details):

- Loading the software from tape into a directory.
- Copying **bfs/oper.*xx*.enet** (where ***xx*** represents the software variable for your Annex) to your **tftp** directory.
- Creating the configuration file in your **tftp** directory.

Multiple Server Hosts

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 (see the *Annex Software Installation Notes* for more details).

Parsing the Configuration File

The configuration file contains Annex configuration information. It resides on the preferred boot host and is loaded during the Annex booting process.

The configuration file is parsed one line at a time during the booting process. File entries are grouped into sections. Each section begins with a percent symbol (%) followed by a defining keyword.

The configuration file can also contain **include** statements. The **include** statement incorporates separate file entries for backward compatibility.

In earlier software releases, **gateways**, **rotaries**, and **macros** are separate files. Beginning with Release 7.0, the keywords **gateway**, **macro**, and **rotary** are entries in the configuration file. The file syntax has not changed. You can use **include** statements in the configuration file to incorporate files from earlier releases.

You can define a file name for the configuration file maintained on the load host using the Annex parameter **config_file**. The default file name is **config.annex**.

If the Annex is configured for self booting, the configuration file must reside in the Annex's root directory.

File Sections

Each section of the configuration file begins with a keyword preceded by the percent symbol (%). The keywords are: **gateway**, **macro**, **modem**, **rotary**, **dialout**, and **service**. The syntax is:

%*keyword*

You must specify the keyword before defining each section entry; otherwise, a format error occurs. The keyword distinguishes one section of the configuration file from another for the parser. All the statements under the **%**keyword should belong to the same section, including the statements in the **include** file.

Include Statement

The **include** statement tells the parser that entries in the file specified in the *filename* field are part of the configuration file. The syntax is:

%*keyword*
%include *filename*

Setting Up the Configuration File

The following sample configuration file defines the **gateway** entry first. This entry includes a separate file named **test.route** followed by **macro**, **rotary**, **dialout**, and **service** entries. The **rotary** entry includes another file named **test.rotary**.

```
#
#The followings are definitions of the gateways entries
#
%gateway

net 129.91.0.0 gateway 132.245.1.1 metric 1 hardwired
net 129.122.0.0 gateway 132.245.1.1 metric 1 hardwired

annex 192.9.200.228
net 129.123.0.0 gateway 132.245.2.1 metric 1 hardwired
snmp contact crow@xenna
snmp location Room without a view
end

# the file test.route is also part of configuration
%include test.route

# Configure SNMP
snmp community public
snmp traphost 192.9.200.95
```

(continued on next page)

```
#
# The followings are definitions of the macro entries
#
%macro

alias  /Show users on the network/
     keyin “3” 1-6,10-16,v@opus
     keyin “4” 1-6,10-16,v@opus
{
<who
<pause
}

menu  |ANNEX MENU ONE|
     keyin “menu1” 1-16,v@opus
     init_cli 6,v@opus
     cmd_list bg,db_mgr,fg,hosts,jobs,rlogin

{
    MY MENU SCREEN
Choices are:

bg db_mgr fg hosts jobs and rlogin

command:
}

# service entries
%service

service adm_modem\
identification ‘system administrator modem’\
password anypasswd3453 ports 5\
connections enabled queue disabled

service printer\
identification ‘3rd floor laser printers’\
ports 8,11-12\
connections enabled queue enabled

end
```

(continued on next page)

```
# rotary entries

%rotary
%include test.rotary

# All consoles from rack annexes

titanic_co:   2@192.9.200.232
brazil_co:    5@192.9.200.232
botswana_co: 6@192.9.200.232

# Table top annex consoles

zinc_co:         25@192.9.200.230
total_recall_co: 26@192.9.200.230
conan_co:        27@192.9.200.230

# Remote Annex (192.9.200.247) - titanic

titanic_1:            17@192.9.200.232
titanic_2:            18@192.9.200.232

# Remote Annex (jdc)(192.9.200.249) - rlogin

rlogin_1:            protocol=rlogin 8-12@jdc
rlogin_2:            1,9@jdc+132.245.33.229/513
rlogin_3:            protocol=rlogin 13-
16@jdc+132.245.33.228

# dial-out route for jupiter

%dialout

begin_route 5
local                192.9.200.233
remote               192.9.200.234
mode                 slip
ports                6@192.9.200.230
phone                2522555
chat                 chat3
filter               out incl proto icmp disc
disabled             8:00am-6:00pm Mon-Fri
advertise            y
set                  slip_mtu small slip_tos Y
set                  rip_horizon split
end_route

begin_script chat3
send "Slip\r"
end_script
```

Another sample configuration file containing four include statements follows:

```
# The followings are definitions of the gateways entries
#
%gateway
%include gateways
#
# The followings are definitions of the macro entries
#
%macro
%include macros
#
# The followings are definitions of the rotary entries
#
%rotary
%include rotaries
#
# The followings are definitions of the LAT service entries
#
%service
%include service.lat
```

Creating gateway Entries in the Configuration File

You can create **gateway** entries using any text editor. After an Annex boots, it downloads the information from the preferred load host. If the Annex does not locate the configuration file, it assumes the file does not exist on the network. Table A-37 describes the supported keywords for **gateway** entries.

The **gateway** entries configure routes for SLIP and PPP links and enables the Simple Network Management Protocol (SNMP) agent. The **gateway** entries must conform to the following conventions:

- A number 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.) is treated as a single space.
- All keywords and port information are case-sensitive.

A **gateway** entry for a route in the configuration file can have one of the two equivalent formats shown below. The first format is preferred. The second is allowed for backward compatibility with Annex releases prior to R9.3.

Format 1:

route add [**–h**] {*addr1 subnet_mask* | *addr1/bits* |**default**} *addr2* [*metric*]

Format 2:

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

domain search *pathname pathname...*

domain default *pathname*

Table A-36. Supported Keywords for gateway Entries – Format 1

Keyword	Definition
route	The gateway entry is for a host, network, or default route.
add	The route is to be added to the configuration file.
-h	This route is hardwired; it cannot be changed or deleted, even if a routing update is received.
addr1	The route's destination address.
subnet_mask	The subnet mask for *addr1*. If the route is not a default route, you must specify a subnet mask, either in this field or in the /*bits* field (see below).
/*bits*	The subnet mask for *addr1*. This number represents the combined number of 1 bits in the network and subnet mask, from left to right. If the route is not a default one, you must specify a subnet mask either in this field or in the *subnet_mask* field (above). You cannot specify /*bits* for default routes.
default	This is the default route for the Annex.
addr2	The IP address of the next gateway the Annex uses to get to the destination address.
metric	The cost of using this route; typically, this is the number of hops from the Annex to *addr2*.

Table A-37. Supported Keywords for gateway Entries – Format 2

Keyword	Definition
net|host	The destination specified by *addr1* is a network or a host.
addr1	The IP address of the destination; use either 0 or default to add a default route.
/*bits*	This field specifies a subnet mask for *addr1*. The value of *bits* represents the combined number of 1 bits in the network and subnet mask. You cannot specify this field for default routes.
addr2	The IP address of the gateway that reaches the destination address.
hops	The number of hops needed to reach the destination.
active	The gateway can generate RIP update messages. If the 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 active gateway entry.
passive	The gateway cannot generate RIP messages, and therefore, is never removed from the routing table. If a better route is received for the destination, the entry in the routing table is updated.
hardwired	The gateway has a fixed route to the destination. This route cannot be changed or deleted, even if a routing update is received.

(continued on next page)

Table A-37. Supported Keywords for gateway Entries – Format 2 (continued)

Keyword	Definition
domain search	Adds a network domain to the DNS search path. The given paths are decomposed into higher-level names, and all of the forms are added.
domain default	Sets the default search path. This path should be set after all other search paths are added. The Annex propagates this path to the top of the search list and removes it from all of the *hosts* entries.

The CLI **hosts –n** command displays the domain search and the domain default list contents along with the name server data.

Gateway Extensions

The **gateway** extensions allow you to define lines in the file that refer only to a specific Annex, or to all Annexes, or to a specific subnet.

The syntax for an extension that includes a specific IP address is:

annex *ipaddr*
...
end

The syntax for an extension that matches all Annexes (useful for local files and to force route caching) is:

annex *
...
end

The lines enclosed by the **annex...end** block are to be used only by the Annex with the IP address *ipaddr*. Any routes enclosed by the **annex...end** block are cached. An **else** keyword can also 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** blocks.

The following sample extension to a **gateway** entry in the configuration file shows how to configure a SLIP interface (see . *SLIP Link with Two IP Addresses* on page A-140).

```
%gateway
# SLIP link to the 132.245.5 net
annex 132.245.5.9

   # 132.245.10.7 is a gateway to the entire 132.245.5 net
  net 132.245.5.0/24 gateway 132.245.10.7 metric 1 hardwired

else

   # other Annexes will route to 132.245.5 via 132.245.5.9
  net 132.245.5.0/24 gateway 132.245.5.9 metric 2 hardwired

end
```

An extension that matches all Annexes on a given network or subnet is particularly useful for defining a default route shared by these Annexes. The syntax is as follows:

subnet *ipaddr*
...
end

The lines enclosed by the **subnet...end** block are to be used only by Annexes on the subnet or network with the IP address *ipaddr*. Any routes enclosed by the **subnet...end** block are cached. An **else** keyword can also be used (alone on a line) to list configuration information for all subnets/ networks except the one identified on the **subnet** line. You cannot nest **subnet...end** blocks.

The following are sample **subnet...end** blocks:

```
subnet 132.245.33.0
route add default 132.245.33.22 1
end
subnet 132.245.66.0
route add –h default 132.245.66.22.1
end
```

In the sample above, the first default route applies to all Annexes on subnet 132.245.33.0. The route is not hardwired, so it can be replaced by a default route learned by RIP or defined elsewhere. The second default route will never be replaced (unless you change it) because it is specified as hardwired (–h).

The Annex logs errors in **gateway** entries if *syslogging* is enabled. For more details on event logging, see *Encrypting Security Messages* on page A-436. The **syslog_mask** parameter determines the priority levels for logging these event messages (see *syslog_mask* on page C-107).

For information on SLIP links, see *Serial Line Internet Protocol (SLIP)* on page A-137.

For information on PPP links, see *Point-to-point Protocol (PPP)* on page A-111.

For information regarding keywords for configuring the SNMP agent, see *SNMP Management Stations* on page B-42.

Loading the Host Table from the Configuration File

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

A host name entry has an IP 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 sample 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
```

Host name entries may be conditional with the use of the **annex...end** blocks (see *Creating gateway Entries in the Configuration File* on page 14-350). The address on the **annex** statement must be an IP Address. Each IP address has only one associated host name (see *Managing the Host Table* on page B-34).

Routing Services and gateway Entries

The Annex parameter **routed** determines whether or not the Routing Information Protocol (RIP) routing daemon is enabled (see *routed* on page C-101 for more details). When the daemon is enabled, the Annex performs both active and passive RIP routing. If the daemon is disabled, no RIP routing occurs (for more information on RIP, see *IP Routing* on page A-169).

A routing table maintains information on gateways that provide routes to hosts on different networks. If the **routed** parameter is set to **Y**, the Annex builds this table dynamically by monitoring RIP broadcast messages; both Version 1 and Version 2 of RIP messages are accepted. If **routed** is set to **N**, the Annex builds the table by monitoring only ICMP redirects.

The CLI **netstat –r** command displays the routing table.

Another option for maintaining the routing table is to create **gateway** entries in the configuration file for the Annex in which you define fixed routes to a destination. If you disable RIP, the Annex relies only on the **gateway** entries and ICMP redirects.

Route Cache

The route cache contains user-configured routing information (static routes, and sometimes a default route). The Annex copies these routes to the route cache from **annex...end** blocks in the **gateway** section of the Annex configuration file (see *Creating gateway Entries in the Configuration File* on page 14-350) and/or from routes defined via the CLI superuser **route** command (see *Defining Routes* on page A-194).

Routes in the cache include those whose next hops are directly reachable – that is, up and running on a network directly connected to the Annex – and those that are not yet reachable. Routes whose next hops are reachable are immediately copied to the RIP and kernel routing tables. Routes whose next hops are not yet directly reachable are copied to the RIP routing table as soon as their next hops become reachable. The latter technique saves the Annex the trouble of consulting the configuration file, which is typically not stored on the Annex, each time a route's status changes. A copy of the route remains in the routing cache, in case its next hop again becomes unreachable.

The **netstat –C** command displays the route cache (see *Route Cache Information* on page B-17 for more details on using this command).

Disabling RIP

Another option for maintaining routing tables is to disable RIP on the Annex by setting the Annex parameter **routed** to **N**. This prevents the Annex from reacting to RIP broadcasts and using alternate routes. The Annex constructs its routing table from the information contained in the **gateway** section of the configuration file and
ICMP redirects.

If you disable RIP, define a default route to a smart gateway in the **gateway** section of the configuration file. This gateway sends ICMP redirects to the Annex, allowing it to learn only the required routes.

Creating macro Entries in the Configuration File

You can customize the CLI user interface by adding **macro** entries to the Annex configuration file. Using **macro** entries, 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. Table A-38 describes the supported keywords for **macro** entries.

The maximum number of macros per Annex is 128.

The Annex loads the **macro** entries when it boots or when the **reset annex macros** command is issued using either **na** or **admin**. If the preferred load host is not available, and the Annex **load_broadcast** parameter is set to **Y**, the Annex broadcasts for the configuration file.

The **macro** entries in the configuration 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.
- The sequence of backslash-closing brace (**}**) allows a closing brace character (**}**) to appear in the body of a macro.
- Non-delimited white space (i.e., space, tab, etc.) is treated as a single space.
- All keywords and port information are case-sensitive.

- Many strings, including description strings, must be enclosed in delimiters. The delimiters may be any printable character not contained within the string.
- The maximum number of characters allowed between the opening and closing braces is 1024 (a *syslog* message is sent if this length is exceeded).
- Start a macro with an opening brace (**{**) character on a new line.
- End a macro with a closing brace (**}**) character on a new line.

The entries in the **macro** section of the configuration file follow one of these forms:

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

The **alias** begins an alias definition; the **menu** begins a menu definition. The *description* is a string that may contain spaces. The string is the information 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 as:

- 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 IP 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 **macro**. 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.

Sample *port_set* entries look like this:

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

The Annex does not support subsets of virtual connections.

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*. Table A-39 describes the statements permitted in an *alias expansion* on a CLI line.

Table A-38. Supported Keywords for macro Entries

Keyword	Description
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, adaptive, or virtual CLI ports.

(continued on next page)

Table A-38. Supported Keywords for macro Entries (continued)

Keyword	Description
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**. Use only with the **alias** macro form.
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.

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

Table A-39. Statements Permitted in an Alias Expansion

Statement	Description
> *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 messages to the CLI, but not jobs.
< *string*	Simulates the *string* being input from the serial port; for a virtual CLI, it simulates input to the CLI, e.g., CLI commands.
< 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. To define the wait time, use the syntax < *pause xx.x* where *xx.x* is time in seconds and tenths of seconds.

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. In this case, the Annex will display the default Annex prompt. If you include any superuser CLI commands in a menu, they display only when a superuser accesses the menu.

Examples of Aliases and Menus

The following sample **macro** entries define a series of aliases combined into a menu. The menu is accessible from port 3 and all virtual CLI connections for an Annex at a given address.

```
%macro
#
# 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” 1-64@annex-address
#  ( 1-64@address = ports 1 through 64 for annex @ address)
#
#  keyin “2” v@annex-address
#  ( v@address   = all virtual ports for annex @ address)
#
#  keyin “3” v
#  ( v           = all virtual ports for any annex that
#                  boots from this host)
#
#  keyin “4” 1-64
#  (1-64        = ports 1 through 64 for any annex that
#                  boots from this host)
#
```

(continued on next page)

```
# 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.
#
# 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 IP address
#      (e.g., 192.9.200.1).
#
#      Replace system-address with your system's name or IP
#      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
}
```

(continued on next page)

```
#
# Issue a “who” command to determine who is running on the
# Annex; wait for a <Return> before returning to the menu.
#
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 VT100-type term commands
# 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^[[1mEnter Number: ^[[m
}
```

The last entry creates the display for the menu and includes the above aliases in the menu’s **cmd_list**. 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:

```
          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** entries automatically connect 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.

```
alias “Connecting to host”

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

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.

Adding Control Characters

Adding control characters in macros for cursor movement is editor- and system-dependent:

- In the built-in Annex editor, *vi*, *ex*, or *ed*, press CTRL-V followed by the control character you want to insert (e.g., CTRL-[for escape or CTRL-M for a carriage return).
- In *emacs*, press CTRL-Q followed by the control character you want to insert.

For more details or information on other editors, please see your editor’s documentation.

The following sample macro inserts carriage returns along with the line feeds:

```
alias “Connecting to host”
#
# Where the ^M is a carriage return 0xb character.
#
# Menus for logins to various locations
#
#
alias "Aqua"
  keyin "1" 1-64@annex-address
{
<rlogin aqua
}

alias "Peach"
  keyin "2" 1-64@annex-address
{
<rlogin peach
}
alias "Node 7"
  keyin "3" 1-64@annex-address
{
<telnet node7
}

alias "Exit"
  keyin "4" 1-64@annex-address
{
<kill
<hangup
}

menu | Administrative Terminal Service |
  init_cli 1-64@annex-address
  keyin "menu" 1-64@annex-address
  cmd_list 1,2,3,4
{^M
^M
^M
^M
```

(continued on next page)

```
Administrative Terminal Service^M
===============================^M
^M
^M

        1) Aqua
        2) Peach^M
        3) Node  7^M
        4) EXIT^M
^M
Enter selection for desired application (1 - 4) : ^M
}
```

Managing macro Entries in the Configuration File

The CLI **help** command displays an alias name and description with other valid CLI commands. The superuser CLI **help –m** command 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       1,12                     :Menu 1
2       1-16,v                   :Read EMAIL
3       1-16,v                   :Command disabled
4       1-16,v                   :Another who command
==========================================================
==
init_cli 7                       :Set stty commands
init_cli 2,4,6,8,10,12,14,v      :Another who command
init_cli 1,3,5,16                :Dedicated port macro
init_psr 2,4,6,10                :Port 10 information
annex01#
```

If you issue the name defined with the keyword **keyin** along with **help –m**, the command displays the definition defined with that entry:

```
annex01# help –m 2

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

<rlogin maildrop
<mail

<<
annex01#
```

The **reset annex macros** command instructs the Annex to reload **macro**. Thus, you can modify the **macro** section of the configuration file and load it onto an Annex without having to reboot.

Creating service Entries in the Configuration File

The **service** entries in the configuration file define the LAT services that the Annex advertises. Table A-40 defines the available arguments. The syntax for each entry is:

service *name* [**identification** *id_string*] [**password** *password*] \
ports *port_range* [**connections** ***enabled***|***disabled***]\
[**queue** ***enabled***|***disabled***]

Table A-40. Supported Arguments for service Entries

Argument	Description
name	A 16-byte ASCII string defining the service name.
id_string	An optional 32-byte ASCII string containing additional information about the service. The default is an empty string (no identification).

(continued on next page)

Table A-40. Supported Arguments for service Entries (continued)

Argument	Description
password	An optional 16-byte ASCII string defining the service password. The default is an empty string (no password).
port_range	A list of ports offering a particular service. The *port_range* must be integers separated by either a comma or a hyphen. Blanks are not permitted as delimiters.
connections	An optional parameter that defaults to ***enabled***.
queue	An optional parameter that defaults to ***disabled***.

In the following sample **service** entry in the configuration file, the first entry defines a printer service with three printers attached to Annex ports 8, 11, and 12. In this entry:

- Password protection is disabled, and service connection requests are enabled.
- If the user does not specify a port, the request is sent to the first available printer. If all printers are busy, the request is queued for later processing.
- If the user specifies a particular port, the request is sent to that port if it is available or queued until it becomes available.

The next sample **service** entry defines a service called *adm_modem*. The identification field provides additional information about the service. This service is attached to Annex port 5; it is password protected and it is enabled for connection requests. Request queueing is disabled.

The last sample service entry prevents the Annex from advertising VCLI service.

```
% service
annex 132.0.0.50

service printer\
identification ’3rd floor laser printers’\
ports 8,11-12\
connections enabled queue enabled

service adm_modem\
identification ’system administrator modem’\
password anypasswd3453 ports 5\
connections enabled queue disabled

service vcli no

end
```

When queueing is enabled, the service is not advertised and is not visible through the keyword **service**. Also, these services require a special set-up on the LAT host (see *Host Initiated Connections* on page 14-418).

The following sample VMS command file directs a printer queue to an Annex port.

```
$ run sys$system:latcp
create port lta901: /log
set port lta901: /application /node=server_name
                 /service=adm_modem - /port=port_number
exit

$ set terminal lta901: /perm /device=la36 /width=80 /pages=66
-
                       /lowercase /nobroadcast
$ set protection=(s:rwlp,o,g,w:rwlp) /device lta901:
$ set device lta901: /spooled=(queue_name, sys$sysdevice:)

$ initialize /queue /start /processor=latsym /retain=error
      /on=lta901: - /default=(noburst, flag=one, notrailer)
      /record_blocking queue_name
```

To print a file from a VMS host, use the print command:

```
$ print  /queue=queue_name file_name
```

Creating modem Entries in the Configuration File

Each entry in the **modem** section of the configuration file describes the characteristics of a modem connected to the Annex. These include the modem type, related AT commands, commands to reset, dial and set up characteristics, and status values returned by the modem. This information is loaded into the Annex at the initial system boot and after a **reset** command. The Annex saves this information only if the modem type is referenced by at least one port’s **type_of_modem** parameter. Table A-41 lists the field definitions for **modem** entries in the configuration file.

Several standard entries for the **modem** section of the Annex configuration file are supplied with the software distribution. These entries, defined for use with the configuration file, are located in the file **/usr/annex/bfs/modems.annex**.

If the modem you are using is contained within this file, using the **include** *filename* command tells the parser that entries in the specified file are part of the configuration file. For example:

```
%modem
%include modems.annex
```

If the modem you are using is not contained within this file, you must edit the configuration file accordingly.

For more information on modems, see *Modems* on page A-99. For more information on setting parameters, see *Configuration Parameters* on page C-33.

A typical entry for the **modem** section of the configuration file looks like this:

```
%modem
# Acme Widget and Modem Corp Type 1234 V.34 modem

type_of_modem      ACME_288
ready_status       0
connect_status     1 5 10 13 18 20 21 25 43 47 85 91 99 103 107
connect_abort      4 6 12
connect_retry      3 7 8
connect_ignore     11
reset_cmd          AT&F1
reset_delay        3.0+0.0
dialout_setup_cmd  ATE1Q2V0X6&A0&B1&C1&D2&H1&I0&K1&R2S0=0
dialin_setup_cmd   ATE1Q2V0X6&A0&B1&C1&D2&H1&I0&K1&R2S0=1
dial_cmd           ATDT
timeout            60
retry              3
dtr_down_time      60
end
```

Table A-41. Field Definitions for modem Entries

Field	Definition
type_of_modem	This string, containing up to 16 bytes, defines the modem type. This string should match the **type_of_modem** port parameter. This field, which should always appear as the first entry, is required.
ready_status	Defines the *ok* numeric status string the modem returns when a command is successfully executed and the modem is in a ready state. See your modem manual for the applicable *ok* numeric status string. This field is required.

(continued on next page)

Table A-41. Field Definitions for modem Entries (continued)

Field	Definition
connect_status	This string is used only for outbound connections when the Annex is initiating a call. It contains a list of all possible successful *connect* messages, separated by commas, that the modem can return. The Annex accepts up to 80 numeric status words in one or more *connect_status* definition lines. This field is required. If the modem sends a *connect_status* message and asserts DCD, the Annex assumes success, exits the dialer, and starts the protocol or CLI process.
connect_abort	This string contains a list of all possible *modem unusable* messages, separated by commas, that the modem can return (e.g., a *no dialtone* response). This field is optional. If the modem sends a *connect_abort* message, the Annex exits the dialer with a fail and goes to the next modem, if available. If there is only one modem, the Annex resets it and retries the dial. The *retry* field defines the number of retries.
connect_retry	This string contains a list of all possible *dial failure* messages, separated by commas, that the modem can return (e.g., a *busy* response). This field is optional. If the modem sends a *connect_retry* message, the Annex exits the dialer, resets the modem, and retries the dial. The *retry* field defines the number of retries.
connect_ignore	This string contains a list of all possible *ignore these codes* messages, separated by commas, that the modem can return (e.g., a *no dialtone* response). This field is optional. If the modem sends a *connect_ignore* message, the Annex behaves as if the modem has not returned a result code: it does not abort the dial or reset the modem; it does not assume success or failure.

(continued on next page)

Table A-41. Field Definitions for modem Entries (continued)

Field	Definition
reset_cmd	This string, containing up to 16 bytes, resets the modem to a known state (typically the factory defaults). The *reset_cmd* is sent only once if a *ready_status* is received. The modem initialization is not terminated if a *ready_status* is not received in response to the *reset_cmd*. This field is required.
reset_delay	Defines the delay time between sending the *reset_cmd* and receiving the *ready_status*. The time is specified in seconds; valid values range between 0.0 and 25.5. The decimal part is optional, but if used, only one decimal place is allowed. Entries of greater than 25.5 seconds will be interpreted as 25.5 seconds. If this entry is omitted, the default is 3 seconds. This field is optional. The syntax is: reset_delay x.x+y.y where: x.x is the reset completion delay which is pre-empted by a success result code; the default is 5 seconds. y.y is the post reset completion delay which is not pre-empted; the default is zero. This syntax is useful for modems that return success before completing the reset.
dialin_setup_cmd	Sends a setup command to inbound modems before the first user connects to the port. It should include all modem steps required to answer a call coming into the Annex, e.g., it should turn on auto-answer, disable inbound modem connect messages, etc. This field is required.

(continued on next page)

Table A-41. Field Definitions for modem Entries (continued)

Field	Definition
dialout_setup_cmd	Sends a setup command to all slave ports before a port is opened (via telnet, callback, dynamic dial, etc.). This string should contain all of the configuration information required for the Annex to initiate an outbound call, e.g., it should disable auto-answer, enable modem connect messages, etc. If a *ready_status* is not received from the modem, the dialout is aborted. You can specify this string in one or more *dialout_setup_cmd* definition lines, but the combined length of this command must be less than 80 characters long. This field is required.
dial_cmd	A string containing up to 16 bytes that defines the dialout command (typically ATDT). This command returns one of the *connect_status* bytes once the connection is established. See your modem manual for more details on modem commands. This field is required.
timeout	After sending the setup command to the modem, the Annex expects the modem to send back a status byte. This *timeout* value specifies how long the Annex waits for the status byte before assuming that something went wrong, in which case the dial-back is aborted. The default is 60 seconds. This field is optional.
retry	Defines the number of dial-out retries; the default is 3. This field is optional.
dtr_down_time	Holds the DTR signal low for the specified number of seconds. Values range from 1–255 seconds; the default is 4 seconds. This field is optional.
end	Indicates the end of the **modem** entry. This field, which should always appear as the last entry, is required.

Managing Modems

At the initial system boot, the Annex initializes the list of modems in the configuration file. It scans the **modem** section of the configuration file looking for the modems that are defined in the Annex port parameter **type_of_modem**. The Annex loads this modem information into its internal modem table; it stores only the names of other modem types in the configuration file.

You can reset and reload the internal modem table via SNMP or by executing a **reset annex modem_table** either from the **na** utility or the CLI **admin** command:

```
root@host# na
Annex network administrator Rx.x
command: annex 132.245.6.40
command: reset annex modem_table
command: quit
root@host#
```

```
annex# admin
Annex administration Remote Annex Rx.x
admin: reset annex modem_table
admin: quit
annex#
```

The superuser CLI **modem** command lists the modem types that are active in the Annex (for more details, see *modem* on page C-155).

Modem Settings for Dynamic Dialing and Dial-back

The Annex expects the modem to have the following set-up for dynamic dialing or dial-back:

- Use CTS/RTS flow control.
- Have the DCD signal follow the remote Carrier signal.
- Hang up, disconnect, and reset on loss of DTR.
- Have a serial interface rate of 115200 baud, 8 data bits, 1 stop bit, and no parity.
- Force DSR always on.
- Have a fixed DTE speed (automatic speed buffering).
- Numeric result codes enabled (verbose turned *off*).
- Echo turned *on*.
- *Quiet* or *quiet on answer* mode is set to return result codes when originating a call but not to return result codes when answering a call.
- Automatically answer on ring.
- Use the highest possible modem-to-modem data rates (with compression).
- Remember all of these settings by saving them to the modem's non-volatile memory.

Typically, the Annex port parameters are set as follows:

- **control_lines**: **both**.
- **need_dsr**: **Y** (the modem must force DSR true: **&S0**).
- **type**: **dial_in.**
- **input_flow_control**: **eia**.
- **output_flow_control**: **eia**.

Creating rotary Entries in the Configuration File

Each entry in the **rotary** section of the configuration file consists of a rotary name and a definition. The syntax 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. Specify the ports as:

- A single number: `1@annex04` or `6@132.245.6.7`
- A list of numbers separated by commas: `1,3,6,7@annex03` or `12,16@132.24.5.6.9`
- A range of numbers separated by a dash: `6-9@annex01` or `2-12@132.245.6.5`
- Any combination of the above: `1,3,7,6-9@annex01` or `2-12,16,23@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 IP address or a name specified in a name server database. Table A-42 describes the options for the *location* argument; Table A-43 describes the supported keywords.

Table A-42. Valid Options for the location Argument

Option	Description
@annex_inet_addr+ alternate_inet_addr	Defines an auxiliary IP address that can also be used to access the rotary.
@annex_inet_addr+ alternate_inet_addr/513	Defines an auxiliary IP address that can also be used to access the rotary. The 513 extension specifies an **rlogin** connection.
@annex_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. A special version of **rlogin**, one that accepts TCP port numbers, is needed to use TCP port numbers with **rlogin**.

Table A-43. Supported Keywords for the location Argument

Keyword	Description
ps=	invisible | visible
	Determines whether or not the port server displays the rotary name when a user connects to the **telnet** port of the Annex’s primary IP address. Rotaries without auxiliary IP addresses or auxiliary TCP port numbers always are **visible**. Raw rotaries always are **invisible**.

(continued on next page)

Table A-43. Supported Keywords for the location Argument (continued)

Keyword	Description
protocol=	telnet | rlogin | tstty | raw | binary Defines the protocol between the port and the device. The default is **telnet**. The keyword **rlogin** defines a rotary that is reachable through the UNIX rlogin protocol (unless otherwise specified, defaults the TCP port number to 513). The keyword **tstty** defines a rotary for use with the TSTTY protocol; provides a data stream with no character processing; it is intended primarily for program access to the rotary. The keyword **raw** provides a data stream with no character processing; it is intended primarily for program access to the rotary. The keyword **binary** causes the Annex to negotiate with the host to operate in **telnet** binary mode in both directions.
direct_camp_on=	never | ask | always Determines how port camp-on is handled for rotaries with an auxiliary IP address or auxiliary TCP port. Camp-on is the process of waiting for the next free port in the rotary if all of its ports are busy when a connection is attempted. The **ask** argument asks the user to camp-on; **always** causes the port server to camp-on automatically; **never** causes the port server to refuse the **telnet** connection if the rotary is full. The default for **telnet** and **binary** rotaries is **ask**; the default for **raw** rotaries is **never**. You must define the alternate IP address or TCP port for the rotary; camp-on applies when using **telnet** to access this alternate address or TCP port.

(continued on next page)

Table A-43. Supported Keywords for the location Argument (continued)

Keyword	Description
select=	first | next Defines the order in which the rotary selects ports. The default keyword, **first**, directs the rotary to select the first available port in the *port_set*. The keyword **next** directs the rotary to keep track of the last available port that was selected, and to start its search from that point.
phone=	A string of up to 32 characters that defines a phone number for the modem to dial.

The simplest form for a **rotary** entry is:

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

The following example defines a rotary called *HostC*. This rotary uses ports 1 through 4 and ports 29 through 32 on *annex04*. A user accessing the port server on *annex04* through a **telnet** command sees:

```
% telnet annex04

Trying...
     Connected to annex04.
     Escape character is '^]'.

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

Enter Annex port name or number:
```

Minimum uniqueness applies to rotary names; in this case, users can enter *HostC*, *Hos*, or *H*. Users also can connect to *HostC* by entering the port number(s).

You can specify a **rotary** entry for a direct port range as well:

```
HostD: 132.245.33.238/4000
```

In this case, to **telnet** to port 5 of 132.245.33.238, you would enter:

```
% telnet 132.245.33.238 4005
```

You can also specify a direct port range like this:

```
HostD: proto=raw 132.245.33.238/8000
```

In this case, to get the raw port 5 of 132.245.33.238, you would enter:

```
% telnet 132.245.33.238 8005
```

Not specifying a port range indicates that the entry is for a direct port range. If *proto=raw* is specified, the direct port range starts at 7001; if *proto=raw* is not specified, the direct port range starts at 5001 (Telnet).

For information on using the **rlogin** command to access rotaries, see *Assigning Auxiliary Rotaries Internet Addresses* on page A-84.

Each **rotary** entry follows 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 (,), semicolon (;), backslash (\), hyphen (-), and pound (#).
- 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.
- Spaces or tabs must delimit keywords.

For a detailed description on customizing, configuring, and using rotaries, see *Rotaries* on page A-81.

Creating dialout Entries in the Configuration File

Each entry in the **dialout** section of the configuration file defines a dial-out route. Table A-44 lists the field definitions for **dialout** entries. The format for a **dialout** entry looks like this:

```
%dialout

global_timeout <time-out value>

# this is a comment line

begin_route          <route id>
local                <local address>
remote               <remote address>
mode                 <SLIP or PPP>
ports                <port set/rotary>
phone                <phone number>
chat                 <chat script list>
filter               <filter command>
disabled             <time interval>
advertise            <Y or N>
set                  <parameter_name setting>
end_route
```

Table A-44. Field Definitions for dialout Entries

Field	Definition
begin_route	Marks the beginning of a dial-route entry and defines the route ID (an integer).
local	The IP address or machine name of the interface's local endpoint. If this optional field is omitted, the Annex's address is used.
remote	The IP address or machine name of the remote endpoint of the interface. This mandatory field must appear once.
mode	The serial protocol to be used, e.g., SLIP or PPP. This mandatory field must appear once.
ports	Specifies the set of physical ports to which this route can be assigned. This field can have an associated port set (e.g., 5–8,12), rotary name (e.g., rotary_1), or a combination of rotary names and port sets (e.g., rotary_1,4–9). If the entry is a rotary name, you must also enter this name in the **rotary** section of the configuration file. This field must appear at least once (i.e., you can enter multiple *ports* fields). The order in which the rotary name and port sets are entered determines the order in which ports are selected when a port must be chosen. The first port entry has the highest selection level and the last entry has the lowest level. If a port re-appears on another *ports* line, the selection level is determined by the port's initial appearance.
phone	A string of up to 32 characters that defines the phone number for the modem to dial.
chat	This string of up to 32 characters defines the name of the script that coordinates communications with the remote side immediately after the phone connection is established. You can enter multiple chat scripts on a single line separated by commas or by using multiple chat lines; the chat scripts are executed in the order in which they appear in the **dialout** entry.

(continued on next page)

Table A-44. Field Definitions for dialout Entries (continued)

Field	Definition
filter	Defines a filter to apply to the dial-out route. You can enter only one filter per line. The filters are executed in the order in which they appear in the **dialout** entry. Using this field is the only way you can add a dial-out filter; you cannot add one via the CLI **filter** command. Moreover, you omit the word **add** and the interface name from the filters you define in the **dialout** entry.
disabled	Specifies when the dial-out route will be disabled automatically. The syntax is: [*time*] [*day*][– *time day*]. *Time* is specified as: *hh:mm*[*am,pm*]. If *am* or *pm* is not specified, 24-hour notation is assumed. If *time* is not specified, the default is all day. *Day* can be a weekday, i.e., Sunday, Monday, etc. or a month and a day. Weekday and month specifications observe minimum uniqueness and are not case sensitive. If *day* is not specified, the default is every day. During the disabled period, the route will not appear in the Annex's routing tables and cannot be activated. An active route that enters its disabled time interval will have its physical port reset and is removed from the routing tables. A disabled dial-out route will appear in the output of the CLI **dialout** command. Dialing into an Annex with a disabled route can activate the route if the remote address is within the dial-out's subnet. For this reason, disabling the route is effective for saving telephone costs but not for providing security. (continued)

(continued on next page)

Table A-44. Field Definitions for dialout Entries (continued)

Field	Definition

disabled (continued)	Sample entries for the *disabled* field are:
	`8:00am Friday - 6:35pm Friday`
	`Wednesday`
	`10:30 Nov 30 - 21:30 Nov 31`
	`Friday - Sunday`
	In the first example, the dial-out route will be disabled at 8:00 a.m. on Friday and not enabled until 6:35 p.m. on Friday. In the second example, the dial-out route will be disabled all day on Wednesdays. In the third example, the dial-out route will be disabled at 10:30 a.m. on November 30 and enabled at 9:30 p.m. on November 31. In the last example, the dial-out route will be disabled on Fridays, Saturdays, and Sundays.
	Setting the *disabled* field will not prevent another Annex from dialing into this Annex via a complementary dialout route.

advertise	Specifies whether or not a dial-out route will be advertised via RIP even if there are no available ports in its rotary. A **Y** enables this field, an **N** disables it. If *advertise* is not specified, the default is **Y**.
set	Specifies the parameter settings that will be applied to the dial-out connection. These settings will override the values in non-volatile memory while the process is alive, but they will not change the actual values in non-volatile memory. The syntax is: **set** [*parameter parameter_value*] Any parameters not specified in the *set* field are determined by the actual non-volatile memory settings; the Annex disregards any duplicate valid parameter settings. Table A-45 on page A-390 1ists the configuration parameters that can be set within this field.
end_route	Marks the end of a dial-out entry.

Table A-45. Parameters that can be Set within the set Field of the dialout Entry

Port Generic Parameters	
autobaud	parity
data_bits	speed
location	stop_bits
mode	
Flow Control and Signal Parameters	
control_lines	input_flow_control
input_start_char	input_stop_char
output_flow_control	output_start_char
output_start_char	need_dsr
ixany_flow_control	backward_key
forward_key	
Port Security Parameters	
port_password	user_name
Serial Networking Protocol Parameters	
allow_compression	net_inactivity
dialup_addresses	net_inactivity_units
do_compression	phone_number
local_address	remote_address
metric	slip_ppp_security
SLIP Parameters	
slip_allow_dump	slip_no_icmp
slip_load_dump_host	slip_tos
slip_mtu_size	subnet_mask

(continued on next page)

Table A-45. Parameters that can be Set within the set Field (continued)

PPP Parameters	
ppp_acm	ppp_security_protocol
ppp_mru	ppp_username_remote
ppp_password_remote	
Interface Routing Parameters	
rip_accept	rip_recv_version
rip_advertise	rip_send_version
rip_default_route	rip_sub_accept
rip_horizon	rip_sub_advertise
rip_next_hop	

It is unnecessary to provide explicit values for the port security parameters **port_password** and **ppp_password_remote**. If an asterisk (*) appears as the value for either of these parameters, and the Annex parameter **enable_security** is set to **Y**, the ACP security server provides the password by searching the **acp_userinfo** file (see *Dial-out Passwords* in the following section).

Dial-out Passwords

A user entry in the **acp_userinfo** file can specify a dial-out password. The keyword *dyndial_passwd* marks these entries, which can appear anywhere in the user record. ACP uses the *dyndial_passwd* if the configuration file's *dialout set* field for either *port_password* or *ppp_password_remote* contains an asterisk (*) and the **enable_security** parameter is set to Y. (For more details on using the **acp_userinfo** file, see *Creating the acp_userinfo File* on page A-454):

```
user smith
     dyndial_passwd   jupiter
end
```

If the configuration file's *dialout set* field for either *port_password* or *ppp_password_remote* contains a valid value other than *, the Remote Annex uses that value. If neither parameter is set in the configuration

Dial-out Routes

After booting or executing the **na**/**admin** command **reset annex dialout**, the Annex loads each dial-out route belonging to it and assigns each route an interface name. Each dial-out interface name is in the form **do** *route_id* (the *route_id* is specified in the *begin_route* entry in the **dialout** section of the configuration file). You can configure the Annex for any number of dial-out routes and the *route_id*s do not have to be contiguous. The dial-out route interface names can change only after a system reboot or a **reset annex dialout** command.

The Annex ignores any dial-out route that has a mode (SLIP, PPP) that is disabled via the **disabled_modules** parameter.

Chat Scripts

Chat scripts coordinate the communications between each side of a dialed connection. They consist of a sequence of exchanges, usually following the pattern: *send x, expect y, receive z, compare y and z. If OK, continue with the next exchange*.

A common application for a chat script is the login sequence. When dialing into a host that is running a login process on that serial port, a sequence of username/password prompting occurs. A chat script can send the username and password automatically, thereby logging the user into the port. Chat scripts are also helpful when dialing into the Annex. For example, a chat script can start a SLIP process on the dialed port by having the **slip** command in the script.

A chat script is not applicable to all dial-out routes. It is not applicable when the port receiving the call is running SLIP or PPP and waiting to be dialed. It is applicable (and mandatory) for a dial-out route when the dialed port is a dial-in CLI or is running some shell native to the dialed machine.

Chat scripts, like dial-out routes, are configured in the **dialout** section of the configuration file. Each chat entry begins with the field called *begin_script*. Table A-46 defines the field definitions for chat scripts. Table A-47 defines the reserved words used in chat scripts.

The format of a chat script looks like this:

```
% dialout

begin_script          <script name>, <time-out value
(optional)>
call                  <script name>
sleep                 <sleep length>
send                  <string>[,<string>]...
expect                <string>[,<script name>]
expect_case           <string>[,<script name>]
timeout               [<time-out value>][,<script name>]
end_script
```

The *call*, *sleep*, *send*, *expect*, and *expect_case* statements can appear in any order and in any combination.

Table A-46. Field Definitions for Chat Scripts

Field	Definition
begin_script	Marks the beginning of a chat script; specifies the name of the script along with an optional time-out value. Contains a maximum of 32 characters.
call	Executes another chat script.
sleep	Causes a pause in the script for <*sleep length*> in seconds.
send	Sends a string. The string must be enclosed in double quotes. Multiple strings can be entered on one line or on multiple lines.

expect
 	
expect_case
	Sends control to another script upon reception of a string; the first match is used. When <*string*> is received, <*script name*> is called; the default is *continue*. The *expect* field is not case-sensitive. The *expect_case* field is for use with case-sensitive matches. You can block together multiple *expect* statements:
	
	`expect <string 1>, <script 1>`
	`expect <string 2>, <script 1>`
	`expect <string 3>, <script 3>`

timeout	Specifies the total amount of time to wait (in seconds) for strings defined in a block of *expect* statements. The *timeout* statement terminates a block of *expect* statements. The <*script name*> is the chat script name to execute upon timeout; the default is *error.*
end_script	Marks the end of a chat script.

String Formatting Extensions

Any character can be inserted into the *send*, *expect*, and *expect_case* strings using the backslash (\) character followed by an octal number of no more than three characters. For example, to send a newline character (octal: 12), insert a \12 in the *send* string. The sequence of octal numbers following the backslash character cannot exceed 377 (255 decimal). Some control characters can also be represented as follows:

Name	Decimal Value	Representation
BEL	7	\a
BS	8	\b
TAB	9	\t
LF	10	\n
FF	12	\f
CR	13	\r

To send the backslash character (\), insert it into the string using "\\\\". Send a break using "\k".

Reserved Keywords

Table A-47 defines the four reserved keywords that can be used in place of a script name in the *expect*, *expect_case*, and *timeout* chat script statements.

Table A-47. Reserved Keywords Used in Place of a Script Name

Keyword	Definition
success	Stop all chat activity and return successfully. Go to next chat in dialout, if any.
error	Stop all chat activity and return unsuccessfully. Abort dialout. In the *timeout* statement, if <*script name*> is omitted, *error* is assumed.
continue	Continue executing current script. In the *expect* statement, if <*script name*> is omitted, *continue* is assumed.
return	Return successfully from currently executing script.

Table A-48 defines the two reserved keywords that can be used in place of a string in the *send*, *expect*, and *expect_case* statements (see *Chat Script Examples* on page 14-398 for a sample chat script that uses these keywords).

Table A-48. Reserved Keywords Used in Place of a String

Keyword	Definition
user_name	Use the user name associated with the port. Do not put quotes around this entry.
port_password	Use the password associated with the port. Do not put quotes around this entry.

Default Global Timeout Values

The overall time for a dial-out route to complete its chat script activity is two minutes (120 seconds). You can override this value using the *global_timeout* statement. The syntax is:

global_timeout <*global time-out value*>

The *global_timeout* statement must appear inside of the **dialout** section of the configuration file but outside of any route.

Default Timeout Values

If the <*timeout value*> is omitted from the *timeout* statement, the default of five seconds is used. Each chat script can have its own default *timeout* value by specifying a value in the *begin_script* statement using the format:

```
begin_script <script name>, <default time-out value>
```

See *Chat Script Examples* on page 14-398 for a sample chat script that uses this format.

Typically, the *timeout* statement terminates a block of *expect* statements. If a block of *expect* statements is terminated by a statement other than *timeout*, the <*time-out value*> used is the <*default time-out value*> for the script, and the <*script name*> used is *error*.

Chat Script Examples

The following sample chat script illustrates the Annex’s chat script language. This script first calls the chat script called *chat2*. If *chat2* is successful, *chat1* continues (i.e., sleeps for 5 seconds). If *chat2* is unsuccessful, *chat1* returns as unsuccessful. The chat script then sends the string called *String1*. If *String2* (case-sensitive) does not arrive within 5 seconds, an error is returned. If *String2* does arrive within 5 seconds, the chat script called *chat3* is called. Since *chat3* always returns successfully, *chat1* also returns successfully.

```
begin_script        chat1
call                chat2
sleep               5
send                "String1\r"
expect_case         "String2", chat3
timeout             5, error
end_script

begin_script        chat3
send                "String3\r"
end_script

begin_script        chat2
sleep               3
send                "Message #1\n"
expect              "OK"
expect              "NOT OK", error
timeout             10
end_script
```

The following sample chat script can start a SLIP line on the called Annex CLI port. This script sends the string *slip* with a carriage return. Then it waits 5 seconds for a case-sensitive match on the string *Switching to SLIP*. If the match times out, the script will return as unsuccessful. If the *expect_case* field receives the expected string, it returns successfully.

```
begin_script          chat_slip
send                  "slip\r"
expect_case           "Switching to SLIP", success
timeout               5, error
end_script
```

The following sample script waits ten seconds for the string *username*: because this <*default time-out value*> is specified in the *begin_script* statement. The <*default time-out value*> is used only if a <*time-out value*> is omitted from the *timeout* statement. If a <*default time-out value*> is not specified on this line, the default remains five seconds.

```
begin_script          script1, 10
send                  “login”
expect                “username:”
timeout               success
end_script
```

The following sample chat script illustrates the use of the reserved keywords *user_name* and *port_password*:

```
begin_script          script1
send                  “\r\r”, user_name, “\r”
expect                “password:”, continue
timeout               error
send                  port_password, “\r”
expect                “successful”, success
timeout               error
end_script
```

Setting Up the motd 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 same directory that holds the operational images (usually **/usr/spool/erpcd/bfs**). The default file name is **motd**. If you use another name for this file, you must specify this name using the Annex parameter **motd_file**.

Remote Annex Server Tools for Windows NT® stores the **motd** file in: <*product-installation-root-drive*>\bfs sub-directory.

The Annex reads the **motd** file from the file server host each time it boots, and when the **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 by setting the **load_broadcast** parameter to **N**.

Configuring an Annex as a Boot Server

To configure an Annex as a boot server for other Annexes, set the **server_capability** parameter; Table A-49 lists the valid arguments.

An Annex configured with R9.0 and above software requires a minimum of 4 MB of RAM in order to act as a file or boot server.

Table A-49. Arguments for the server_capability Parameter

Argument	Definition
all	The Annex is a file server for the configuration, operational image, and message-of-the-day files.
config	The configuration file.
image	The Annex's operational code.
motd	The message-of-the-day file.
none	The Annex is not a file server. This is the default value.

Self-booting without a Local Ethernet Interface

When booting an Annex from the Flash ROM without a local Ethernet interface:

1. **Enter the ROM Monitor prompt from the console.**
2. **Set the IP address to a valid IP address and the subnet mask to a valid mask using the** addr **command.**
3. **Set the interface sequence to self using the** sequence **command.**
4. **Boot the Annex.**

Using the Annex FTP Daemon

Using the Annex FTP daemon, you can upload or download files (those visible through the superuser CLI **ls** and **edit** commands) in the Annex's non-volatile memory (EEPROM) from a remote host.

The Annex FTP daemon is primarily useful for saving Annex configuration files to a host on the network for the purpose of “swapping” Annexes.

You cannot “get” or “put” boot images using the Annex FTP daemon.

For more details on using the Annex FTP daemon, see *Configuring Security for the Remote Annex FTP Daemon* on page A-542.

Installing a Time Server

The Annex host software includes a time server program. Time servers are available on many UNIX hosts through the **timed** 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 displays as listening on UDP port **time** (or **37**).

Remote Annex Server Tools for Windows NT® ships with its own time server.

If a host time server is not present, use the supplied **timserver** program:

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

```
time 37/udp timserver
```

2. **Start the server:**

```
# /etc/timserver
```

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** and **time_server** Annex parameters.

DNS does not require the **timserver** program.

Dump Host Services

The Annex can dump its memory image to a dump host on demand through either the superuser CLI command **boot –d** or the **na** command **dumpboot** 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 have **bfs** or **tftp** capability. You can define a preferred dump host to which the Annex first tries to up-load a dump file. If this address is not specified or the host is not available, the Annex broadcasts a request and dumps to the first host that responds.

Self -boot units without a network host cannot perform a dump.

At the dump host, the dump creates a file, between one and four Megabytes in size, in the directory **/usr/spool/erpcd/bfs** for **bfs** dumping (in the **tftp** directory for **tftp** dumping), 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 IP 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 IP network address. Subnet addresses are not used in naming the dump file. The name of the dump file is the Annex's IP host address.

If the **pref_dump_addr** parameter is not set and your IP address is corrupted (zero), leave the **Test** button on the Annex's front panel *on* and the unit will prompt for an IP address when it tries to dump.

Table A-50 provides sample dump file names; all pathnames are relative to **/usr/spool/erpcd/bfs**.

Table A-50. 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

Configuring Name Servers

The Annex uses various means of creating and maintaining the host table. This table includes the host names and the corresponding IP 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 IP address.

If broadcasting is enabled, the Annex first makes three attempts for a DNS server followed by three attempts for an IEN-116 server.

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* on page A-31).

By default, the Annex builds the host table exclusively from RWHO broadcasts. Depending on what is available for your network, you can 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 add 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** argument or the **–a** and **–n** arguments. 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**).

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 IP 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 IP addresses.

For example, to obtain an IP 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.
```

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.**

Remote Annex Server Tools for Windows NT® does not support the IEN-116 name server.

To verify that the server responds to queries from the Annex, configure the Annex to use an IEN-116 server (see *Using Name Servers* on page A-31). 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 (see *hosts* on page C-149).

Setting Up a Host for 4.3BSD 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 (for more information on event logging, see *Using Event Logging* on page A-37).

Refer to the *Remote Annex Server Tools for Windows NT*® *User Guide* for information about syslogging in a Windows NT® environment.

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 in **/etc/syslog.conf** 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 (see *Encrypting Security Messages* on page A-436 for more details).

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-specific Annex parameters.

To enable the LAT functions:

1. **Enter the correct** lat_key **parameter value.**
2. **Configure the** disabled_modules **parameter not to include lat.**
3. **Reboot the Annex.**

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 (contact your supplier to obtain a **lat_key** value).

When the administrator selects an Annex via **na** or **admin** that has LAT enabled, the Annex returns the normal configuration line with the string *W/LAT* included:

```
command: annex zippy
zippy: Remote Annex w/LAT Rx.x, 72 ports
```

Advertised Services

Advertised services are the announcements of resources (modems, printers, etc.) that LAT machines have available for use by the network clients. A service announcement carries the name and Ethernet address of the server offering the service, along with a current service rating. Every host that accepts communication sessions is a service provider. All service providers broadcast service announcements periodically (typically, once per minute).

A host 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 host 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.

The **service** entries in the configuration file define the LAT services that the Annex advertises (see *Creating gateway Entries in the Configuration File* on page 14-350 for more details).

Learned Services

Learned services are the services that a LAT machine hears and stores from the network. All services received by a LAT machine are filtered based on the group codes in the service announcement and the access group code of the LAT receiving machine.

Each LAT machine has a group code mask that represents the allowable groups for users of that machine. To store a service announcement, there must be at least one group code common between the advertising machine and the receiving machine.

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.

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** command displays only the services to which Annex users have access.

Restricting Access to LAT Services

The **group_value** parameter specifies which remote group codes can access the local services offered by a particular Annex. To change the status of the Annex’s group codes, the administrator must change the **group_value** parameter.

The **group_value** parameter displays the group codes that are enabled; initially, the value is *none*. Allowable values are 0 through 255. The administrator must determine which LAT services to enable, and set the group codes accordingly. The syntax is:

set annex group_value *group_range* **enable** | **disable**

The *group_range* 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 **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** 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.

Accessing LAT from a Virtual CLI

The **vcli_groups** parameter specifies which remote group codes are accessible to virtual CLI users. All virtual CLI users have the same group code. The syntax is:

set annex vcli_groups *group_range* **enable** | **disable**

Accessing LAT from an Annex Port

The **authorized_groups** parameter specifies which remote group codes are accessible to users on a particular Annex port. Each port has its own **authorized_groups** setting. The syntax is:

set port authorized_groups *group_range* **enable** | **disable**

Reverse LAT

The Annex advertises the services specified in the **service** section of the configuration file. Optionally, the Annex queues host initiated connection (HIC) requests if the requested resource is not available, i.e., the port is in use.

Reverse LAT vcli

To enable the **vcli** service in the Annex, LAT must be configured for the network on which it is to run. Also, the **max_vcli** parameter must be enabled. As long as these conditions are met, the Annex can advertise the **vcli** service to the network.

The advertised service's *services* and *host* fields are set to the value of the **server_name** parameter. The Annex dynamically updates the *rating* field of the service advertisement based on the number of **vcli**s left. If **max_vcli** is set to unlimited, the rating is 255. The **max_vcli** parameter is the total number of **vcli*s*** allowed in the Annex.

When a LAT user connects to the Annex **vcli** service, it is the same as a **telnet vcli**. If all **vcli**s are in use by Telnet and LAT users, the connection request is rejected.

Telnet-to-LAT Gateway

The Telnet-to-LAT gateway enables the system administrator to associate a unique IP address with a specific LAT service. This gateway allows a user to **telnet** to that unique IP address thereby connecting to the associated LAT service.

Before configuring this feature, configure your Annex as described in *Configuring LAT Services* on page 14-409. LAT is operating properly on your Annex if you see your LAT network services appear when you execute the CLI **services** command and you can connect to them using the CLI **connect** command.

To set up the Telnet-to-LAT gateway, the system administrator must add a new entry to the **gateway** section of the configuration file. The syntax for this new entry is:

annex *ip_addr*
        translate telnet *ip_addr* **to lat** *service_name* [**telnet_groups** *group_range*]
            [*host_name* [*port_number*]]
 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 IP address that translates to a LAT service. Both *ip_addr* fields are specified in the standard decimal dot notation.

The *group_range* indicates which remote group codes are accessible for this gateway. The *group_range* is specified as integers (0–255) separated by either a comma or a hyphen. The keyword **telnet_groups** is optional (it allows backward compatibility for users who already have Telnet-to-LAT gateways specified). If **telnet_groups** is not included, all groups are accessible (i.e., 0–255).

The *service_name* is the desired LAT service to which **telnet** connects; the *host_name* is the host that advertises the LAT service; and the *port_number* is the port on the LAT host providing the LAT service. The *host_name* and *port_number* are optional. The *service_name*, *host_name*, and *port_number* can be a maximum of 16 characters. Any errors in the syntax are reported in the **syslog** file.

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 **gateway** section of the configuration file. Violating this rule may 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 **gateway** changes, the Annex specified in the *annex-selector* line must be rebooted. For example:

```
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 previous 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**.

Figure A-29 illustrates the Annex TCP/IP gateway.

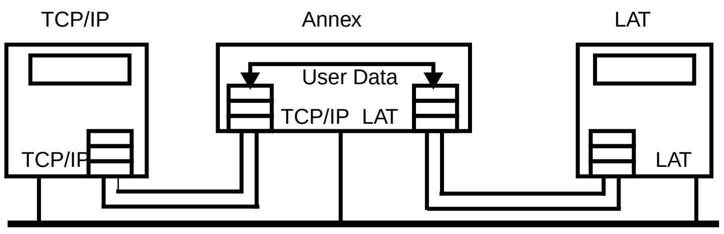


Figure A-29. Annex TCP/IP Gateway

LAT-to-Telnet Gateway

The LAT-to-Telnet gateway allows an Annex to translate an advertised LAT service to a **telnet** connection for a specific IP address. This allows a LAT user to move transparently between the LAT, TCP/IP, and Telnet protocols.

To set up the LAT-to-Telnet gateway, the system administrator must add a translation directive to the **gateway** section of the configuration file. This new entry has the following syntax:

annex *ip_address*
        translate lat *service_name* [**identification** *id_string*] **to telnet**
        ip_address [port_number]
end

For example, a translation entry for a UNIX machine named *frodo* running TCP/IP with the IP address 132.0.0.50 and a VMS host running LAT would look like this:

```
annex 132.0.0.35
     translate lat frodo identification ’Login service for
TCP/IP
     host ’frodo’ to telnet 132.0.0.50
end
```

After making the above translation entry in **gateway**, the system administrator must make sure that the Annex is running LAT and that the **group_value** parameter is configured so that the LAT host can use this service. Also, the Annex must be rebooted.

Using the above sample entry, once the Annex is rebooted, the LAT user will see an entry for *frodo* in the *services* display and the *host* field will correspond to the *service_name* set in the Annex 132.0.0.35. When the LAT user connects to *frodo*, the connection is made through the Annex 132.0.0.35.

Since LAT allows multiple hosts to offer the same service name, many Annexes can offer a LAT-to-Telnet translation to the same TCP/IP host. The hosts that have access to the service will load balance between Annexes, based on the service ratings each Annex advertises for the service.

A **service** entry is not required in the Annex configuration file.

Data-b Slot Support for LAT

The Annex LAT implementation now reports and responds to data-b slot messages. This feature is enabled on a per port basis using the **latb_enable** parameter; the default setting is **N**.

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.

If the **latb_enable** parameter is set to **Y** and the LAT host sends a data-b slot message requesting that flow control (XON/XOFF) be turned off, the Annex turns off flow control and passes XON/XOFF characters to the host. This scenario can adversely affect both XON/XOFF and the terminal's
cursor keys.

Host Initiated Connections

The host initiated connections (HIC) mechanism handles printing from a VMS host. When a VMS host needs to print, it first multicasts on the network. The responding system either accepts the print request or offers to place it in a queue. Once the server can accept the print request, it connects to the host and transfers the data.

Using HIC, it is possible to print to a port that does not have an associated advertised service. All configurations required for standard HIC printing apply, but the user does not have to edit the **service** section of the configuration file, and a service name is not required on the VMS side.

Define the **mode** parameter for the Annex port used for HIC.

The following sample VMS command file sets up a printer queue directed to an Annex port:

```
$ run sys$system:latcp
create port lta901: /log
set port lta901: /application /node=server_name
                 /service=adm_modem - /port=port_number
exit

$ set terminal lta901: /perm /device=la36 /width=80 /pages=66
-
                       /lowercase /nobroadcast
$ set protection=(s:rwlp,o,g,w:rwlp) /device lta901:
$ set device lta901: /spooled=(queue_name, sys$sysdevice:)
$ initialize /queue /start /processor=latsym /retain=error
      /on=lta901: - /default=(noburst, flag=one, notrailer)
      /record_blocking queue_name
```

To print a file from a VMS host, use the print command:

```
$ print  /queue=queue_name file_name
```

Table A-51 defines the variables used in the previous examples.

Table A-51. Variable Definitions for VMS Command File

Variable	Definition
server_name	This name should match the name defined in the LAT-specific Annex configuration parameter **server_name**.
adm_modem	The *system administrator modem* defined in the **service** section of the configuration file.
port_number	The Annex port number associated with the port listed in the **service** section of the configuration file.
queue_name	The VMS queue name.
file_name	The name of the VMS file to print.

Miscellaneous LAT Parameters

The LAT-specific **na** parameters can be changed, but are not necessary to access LAT services; only the **group_value**, **vcli_groups**, and **authorized_groups** parameters are 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.

For more details on using **na** commands, see *na Commands* on page C-1.

For more details on using the CLI commands, see *Using the CLI Commands* on page C-121.

Chapter 15 Using Remote Annex Security

The Remote Annex provides comprehensive security features that assist you in securing your network from unauthorized access. Using these features, you can select between local password protection (where the passwords are stored on the Remote Annex) and host-based security (where at least one host on the network is functioning as a security server).

If unauthorized users can access your Remote Annex, we strongly suggest that you enable security after loading the host code and booting the unit.

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 (for more information, see *Event Logging Using syslog* on page B-29).

Local password protection can be used as a stand-alone security mechanism or as a back-up to host-based security. It validates port access from either the device or the network. Local password protection supports **cli**, **slave**, and **adaptive** ports. The local password protection policy cannot be altered.

The Remote Annex's Access Control Protocol (ACP) provides host-based security in which a UNIX host on the network is defined as a security server. You can modify the host-based software to implement a security policy that fits the needs of your environment.

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

If the **enable_security** parameter is set to **N**, no security is used, and no logging is performed regardless of any other parameter setting.

The **enable_security** parameter does not take effect until the Remote Annex is either rebooted or reset.

The **enable_security** parameter has no effect on the Novell-based security used with ports set to **ndp** mode.

Overview of Local Password Protection

Local password protection allows you to assign a password that a user must enter before accessing a Remote Annex. Because this password is stored locally on the Remote Annex, it does not require a remote security server. Local password protection can be used as a back-up security mechanism in case the host-based security servers are unavailable.

The passwords set in the following parameters are stored on the Remote Annex and do not involve the use of a security server:

- **password**.
- **port_password**.
- **ppp_password_remote**.
- **vcli_password**.

The following subsections describe local security on the Remote Annex provided that the **enable_security** parameter is set to **Y** and a host based security server is not available.

Implementing Local Virtual CLI Password Protection

Local password protection can be implemented for the Remote Annex in one of two ways:

- Upon virtual CLI (VCLI) connection.
- Upon access through administrative utilities.

The **vcli_password** parameter allows you to define a local password for VCLI connections. The user enters only a password as opposed to a user name and password.

To configure the Remote Annex for a local VCLI password protection:

1. **Enable local security by setting the enable_security parameter to Y.**
2. **Disable VCLI remote security by setting the** vcli_security **parameter to** N**.**
3. **Define a password using the** vcli_password **parameter.**

The Remote Annex acts as follows:

- If the **vcli_password** parameter is not set (**"<unset>"**) and the **vcli_security** parameter is set to **N**, the Remote Annex prompts for the password specified by the **password** parameter.
- If the **password** parameter is not set (**"<unset>"**), the Remote Annex fails the VCLI attempt.
- If the **vcli_security** parameter is set to **N** and the **vcli_password** parameter is set (**"<set>"**), the Remote Annex prompts for the password specified in **vcli_password**.
- If the **vcli_security** parameter is not set (**N**) and the **vcli_password** parameter is not set (**"<unset>"**), the Remote Annex does not perform a security check for VCLIs.

- If the **vcli_security** parameter is set to **Y**, the **vcli_password** parameter is not set (**"<unset>"**), and the **password** parameter is not set (**"<unset>"**), the Remote Annex denies access to the VCLI if the security server is unreachable.

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

To set up the local VCLI password for back-up security:

- Enable security by setting the **enable_security** parameter to **Y**.
- Enable VCLI security by setting the **vcli_security** parameter to **Y**.
- Define a password using the **vcli_password** parameter.
- Define a security server host using the **pref_secure1_host**, **pref_secure2_host**, or **security_broadcast** parameter (e.g., 0.0.0.0).
- Create a password file on the security server (see *Creating User Password Files* on page 15-451).

If the remote server(s) fail:

- Access is permitted only through the VCLI password.
- No access is permitted if the **vcli_password** parameter is not set.

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

Administrative Password

The Remote Annex administrative password protects the administrative tools; the default administrative password is the Remote Annex’s IP address. When the **show annex** command 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 (if the **vcli_security** parameter is set to **N**).

Modifying the assigned administrative password enables password protection on access to Remote Annexes through **na** and **admin**.

The administrative password validates access to a Remote Annex through **na** only when security is enabled and the password is defined. Also, it can be used as the VCLI password and to override the password assigned with the CLI **lock** command.

The administrative password never displays. If you forget the modified password, you can reset it only by erasing the Remote Annex’s non-volatile memory using the ROM Monitor **erase** command, and re-entering all parameters.

As a safeguard against losing the unit’s current configuration, use the **na** command **write** to save the Remote Annex and port configurations; if necessary, you can restore these configurations using the **read** command (for more details on using these commands, see *na Commands* on page C-1).

Protecting the Superuser CLI

A Remote Annex administrative password is required for access to the superuser CLI. The default password is the Remote Annex's IP address. There are two ways to change the password:

- Using the superuser CLI **passwd** command.
- Changing the **password** parameter using **na** or **admin**.

Using either method, the new password takes effect immediately for access to the superuser CLI. Reset the password to the Remote Annex's IP address by either:

- Using **na** or **admin** to set the **password** parameter to the null string:

  ```
  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.

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 Remote 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 Remote 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 na Utility from Unauthorized Access

When using the **na** utility, users can access Remote Annex parameters and obtain useful information, or reconfigure and reboot Remote Annexes. Protecting **na** involves UNIX superuser protection and the Remote Annex administrative password.

Upon installation, **na** is owned by root and executable by all. Only a superuser can execute the **set**, **reset**, **broadcast**, **dumpboot**, **boot**, **read**, and **copy** commands.

Overview of Host-based Security

ACP security has three requirements: 1) at least one UNIX host on the network must act as a security server running Remote Annex security software; 2) security must be enabled on the Remote Annex (the **enable_security** parameter is set to **Y**); and 3) a security regime, such as **acp** or **securid**, must be defined for authenticating Remote Annex users.

The security server maintains a database of files that reside by default in the directory **/usr/annex**. These files include:

- **acp_keys** (encryption key information).
- **acp_dialup** (user names and addresses for dial-up connections).
- **acp_group** (user-group associations for security).
- **acp_regime** (security authentication system and associated password file name).
- **acp_passwd** (security passwords).

- **acp_userinfo** (initial login environment and start-up CLI commands).
- **acp_restrict** (restricted hosts and host ports).
- **acp_logfile** and **acp_logfile.*Annex_IPaddress*** (security audit trails).

The contents of these files should match on all security servers (except for **acp_logfile** and **acp_logfile.*Annex_IPaddress***).

The following sections describe these aspects of ACP security:

- *Basic ACP Configuration* on page 15-429
- *Encrypting Security Messages* on page 15-436..
- *Defining Security Profiles* on page 15-439.
- *Using include Files in the acp_userinfo File* on page 15-477.
- *Specifying Modem Pools Within the acp_userinfo File* on page 15-477.
- *Creating the acp_dialup File* on page 15-481.
- *Using Dial-back Security* on page 15-484.
- *Enhancing Password Security* on page 15-486.
- *Using AppleTalk Security* on page 15-498.
- *Using IPXCP Security* on page 15-500.
- *Using Proprietary IPX Security* on page 15-500.
- *Using PPP Security* on page 15-514.

- *Using Kerberos Authentication* on page 15-522.
- *Using the SecurID Card* on page 15-524.
- *Using SafeWord AS Security* on page 15-534.
- *Configuring Security for the Remote Annex FTP Daemon* on page 15-542.
- *Configuring the IP Basic Security Option (IPSO)* on page 15-543.
- *Logging Security Events* on page 15-545.
- *Modifying the Supplied Security Application* on page 15-546.

Basic ACP Configuration

This section outlines procedures for configuring the basic ACP features and describes what happens in each case if ACP goes down.

CLI Security

You can set up host-based security for CLI connections in which users must provide a valid user name and password before they are granted access to a CLI:

1. **Set the** cli_security **parameter to** Y**, so that the Remote Annex will use ACP.**
2. **Define a security server using the** pref_secure1_host**,** pref_secure2_host**, or** security_broadcast parameter **(see *Configuring the Security Server* on page 15-434).**
3. **Create entries in the** acp_regime **file defining the authentication systems to be used and the conditions under which to use them.**

   The install program creates the **acp_regime** file, then prompts you for a default regime and (in some cases) a password file name, which it enters into **acp_regime**. Subsequently, you can add to and/or change the contents of this file. (See *Configuring the acp_regime File* on page 15-449.)

4. **Create entries in the appropriate password files (see *Creating User Password Files* on page 15-451).**
5. **(Optional) Configure encryption for security messages (see *Encrypting Security Messages* on page 15-436).**

If ACP is down, the Remote Annex acts as follows:

- First, the Remote Annex prompts for the password specified in the **port_password** parameter. If the **port_password** parameter is not set (**“<unset>”**), the Remote Annex does not connect the user to the CLI.
- If the **cli_security** parameter is set to **N** and the **port_password** parameter is set (**“<set>”**), the Remote Annex prompts for the password specified in **port_password**.
- If **cli_security** is set to **N** and the **port_password** parameter is not set (**“<unset>”**), the Remote Annex does not perform a security check for CLI connections and allows unrestricted access to the CLI.

Virtual CLI Security

You can set up host-based security for virtual CLI (VCLI) connections in which users must provide a valid user name and password before they are granted access to a virtual CLI:

1. **Set the** vcli_security **parameter to** Y**, so that the Remote Annex will use ACP.**
2. **Define a security server using** pref_secure1_host**,** pref_secure2_host**, or** security_broadcast **parameter (see *Configuring the Security Server* on page 15-434).**
3. **Create entries in the** acp_regime **file defining the authentication systems to be used and the conditions under which to use them.**

The install program creates the **acp_regime** file, prompts you for a default regime and (in some cases) password file name, and then enters them into **acp_regime**. Subsequently, you can add to and/or change the contents of this file (see *Configuring the acp_regime File* on page 15-449).

4. **Create entries in the appropriate password files (see *Creating User Password Files* on page 15-451).**
5. **(Optional) Configure encryption for security messages (see *Encrypting Security Messages* on page 15-436).**

If ACP is down, the Remote Annex acts as follows:

- First, the Remote Annex prompts for the password specified in the **vcli_password** parameter (see *Implementing Local Virtual CLI Password Protection* on page 15-423).
- If the **vcli_password** parameter is not set (**"<unset>"**) and the **vcli_security** parameter is set to **N**, the Remote Annex prompts for the password specified by the **password** parameter.
- If the **password** parameter is not set (**"<unset>"**), the Remote Annex fails the VCLI attempt.
- If the **vcli_security** parameter is set to **N** and the **vcli_password** parameter is set (**"<set>"**), the Remote Annex prompts for the password specified in **vcli_password**.

Connection Security

You can authorize or deny IP or CLI access to specific hosts, host ports, or networks for a particular user, group, time of day, Remote Annex port, or protocol in use.

1. **Define a security server using the** pref_secure1_host**,** pref_secure2_host**, or** security_broadcast parameter **(see *Configuring the Security Server* on page 15-434).**

2. **Set the** connect_security **parameter to** Y**, so that the Remote Annex uses ACP on a CLI connection (via** telnet **and/or** rlogin**).**
3. **Configure the** acp_restrict **file on the security server (see *Limiting Access to Hosts via acp_restrict* on page 15-472).**

   For CLI **telnet** or **rlogin** connections, ACP checks **acp_restrict** to see whether or not access should be granted to the user. For SLIP and IP over PPP connections, the **acp_restrict** file controls access by creating filters based on your input.
4. **(Optional) Configure encryption for security messages (see *Encrypting Security Messages* on page 15-436).**

SLIP and PPP Security

You configure access to a SLIP or PPP link from the Remote Annex as follows:

1. **Set the** mode **parameter to** cli **and have the user issue the** slip **or** ppp **command from the CLI.**

   If the **mode** parameter is set to **slip**, the Remote Annex does not perform a security check.
2. **If you want authentication performed on the CLI connection (rather than authenticating when the user issues the** slip **or** ppp **command), set the** slip_ppp_security **and** cli_security **parameters to** Y**. Then proceed to step 4.**
3. **To have authentication performed when the user issues the** slip **or** ppp **command (rather than authenticating when the CLI connection is made) set the** cli_security **parameter to** N **and the** slip_ppp_security **parameter to** Y**.**
4. **Define a security server using the** pref_secure1_host**,** pref_secure2_host**, or** security_broadcast parameter **(see *Configuring the Security Server* on page 15-434).**

5. **Create entries in the** acp_regime **file defining the authentication systems to be used and the conditions under which to use them.**

   The install program creates the **acp_regime** file, then prompts you for a default regime and password file name, which it enters into **acp_regime**. Subsequently, you can add to and/or change the contents of this file. (See *Configuring the acp_regime File* on page 15-449.)

6. **Create entries in the appropriate password files (see *Creating User Password Files* on page 15-451).**

7. **(Optional) Configure encryption for security messages (see *Encrypting Security Messages* on page 15-436).**

If ACP is down, the **slip** or **ppp** command fails.

The Remote Annex never uses local security with the **slip** or **ppp** command.

Port Server Security

You can set up security for port servers in which users must provide a valid user name and password before they are granted access to an outgoing port:

1. **Set the** port_server_security **parameter to** Y**, so that the Remote Annex will use ACP.**

2. **Define a security server using the** pref_secure1_host**,** pref_secure2_host**, or** security_broadcast parameter **(see *Configuring the Security Server* on page 15-434).**

3. **Create entries in the** acp_regime **file defining the authentication systems to be used and the conditions under which to use them.**

   The install program creates the **acp_regime** file, then prompts you for a default regime and password file name, which it enters into **acp_regime**. Subsequently, you can add to and/or change the contents of this file. (See *Configuring the acp_regime File* on page 15-449.)

4. **Create entries in the appropriate password files (see *Creating User Password Files* on page 15-451).**
5. **(Optional) Configure encryption for security messages (see *Encrypting Security Messages* on page 15-436).**

If ACP is down, the Remote Annex acts as follows:

- If the **port_server_security** parameter is set to **Y**, the Remote Annex prompts for the password specified in the **port_password** parameter.
- If the **port_password** parameter is not set (**"<unset>"**), the Remote Annex fails the port connection attempt.
- If the **port_server_security** parameter is set to **N** and the **port_password** parameter is set (**"<set>"**), the Remote Annex prompts for the password specified in **port_password**.
- If the **port_server_security** parameter is set to **N** and the **port_password** parameter is not set (**"<unset>"**), the Remote Annex does not perform a security check for port connections.

Configuring the Security Server

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

The **erpcd** process must be running; **erpcd** requires the **/etc/services** file to have an entry for *erpc 121/udp*.

Setting Up a Security Server

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

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

Specifying the Security Hosts

The **pref_secure1_host** and **pref_secure2_host** parameters specify the preferred security hosts. The Remote Annex first queries the **pref_secure1_host** for user validation. If a response is not received within the time defined in the **network_turnaround** parameter, the Remote Annex repeats the query several times. If the Remote Annex still does not receive a response, it queries the host defined in the **pref_secure2_host** parameter. If a response is not received from the second security host within the allowable time limit, and the **security_broadcast** parameter is set to **Y**, the Remote Annex broadcasts to the network for another host with **erpcd** running to authorize the access request. If the **security_broadcast** parameter is set to **N**, the Remote Annex denies the authentication request.

The **network_turnaround** parameter specifies the amount of time in seconds in which the Remote Annex expects a response from the security servers. To reduce the possibility of a retry, the network turnaround time should be long enough to allow for a network transmission to the security server and transmission back to the Remote Annex; unfortunately, if this period of time is too long, the Remote Annex will attempt multiple retries before sending a query to the second security server.

Disabling Broadcasting for Security Servers

The Remote Annex broadcasts to the network for a security server if:

- The **security_broadcast** parameter is set to **Y**.
- The **pref_secure1_host** and **pref_secure2_host** parameters do not respond.

Setting the **security_broadcast** parameter to **N** disables Remote Annex broadcasting. If the hosts defined in the **pref_secure1_host** and **pref_secure2_host** parameters do not respond, the Remote Annex refuses the connection request.

Encrypting Security Messages

Messages between the security server and the Remote Annex are encrypted if the Remote Annex parameters **enable_security** and **acp_key** are set. The parameters do not take effect until the Remote Annex is either rebooted or Remote Annex security is reset.

The **acp_key** parameter specifies the encryption key the Remote Annex uses to exchange messages with the security server. The security server maintains the encryption key for each Remote Annex in the **acp_keys** file (see *Creating the acp_keys File* below and *Configuring Hosts and Servers* on page A-343).

The encryption key also validates the security host: the host must know the Remote Annex's ACP key for the Remote Annex to consider the host valid. Without the appropriate key, the Remote Annex denies the user's request even if the host is defined as a preferred security host.

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

Creating the acp_keys File

The security server maintains the encryption key for each Remote Annex in the **acp_keys** file. Each entry in this file contains a list of Remote Annex names or IP addresses separated by commas and an encryption key for those Remote Annexes. The Remote Annex or the list of Remote Annexes and the key are separated by a colon. The order of placement in the file is important, as the file is read sequentially.

When the security server receives an encrypted message from the Remote Annex, the server tries to match that key against the key assigned to the Remote Annex in the file. If no match exists, the Remote Annex and the server cannot communicate.

The syntax rules for the **acp_keys** file are:

- Any part of an IP address in the list can be specified with an asterisk (*).
- A backslash (\) is used to continue a line.
- Any ASCII character except spaces and tabs are valid encryption keys (keys are case sensitive).
- Each key can contain a maximum of fifteen characters.

Remote Annexes with no entries are assumed to have no key set. Since wildcards are valid, some entries in the file may require an explicit “no key” declaration:

```
annex01, annex02:   seKret2
#  131.21 net Annexes have the same key except for 3 Annexes
131.21.2.1, 131.21.2.2:
131.21.1.1: SpeciaL
131.21.*:   Gub-Net
```

In the following example, the first three entries specify *insomniac-1* as the key for the Remote Annex whose IP address is 132.245.6.15, no encryption for the Remote Annex whose IP address in 132.245.6.75, and *Piano* as the key for all other Remote Annex on the 132.245.6 subnet. The last entry specifies *gl12ch* as the key for *annex01*, *annex02*, and *annex03*. Each **acp_key** parameter for the Remote 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,annex02,annex03:gl12ch
```

Changing the value of the **acp_key** parameter on any Remote Annex requires the same change to the **acp_keys** file on the security server. The recommended order for changing the ACP encryption key on a Remote Annex is:

1. **Edit the** acp_keys **file on all security server hosts.**
2. **Change the value of the** acp_key **parameter for all affected Remote Annexes.**
3. **Update the cache by sending the erpcd on all security server hosts a HUP signal with** kill**.**

   ```
   kill –HUP <pid_number>
   ```

4. **Reset the security subsystem for all affected Remote Annexes using the** na **command** reset annex security**.**

Defining Security Profiles

The expedited remote procedure call daemon (**erpcd**) that implements ACP permits you to define different security profiles for different users, groups of users, or for other *connection conditions*, such as the time of day or the date. Specifically, you can use the **acp_regime**, **acp_userinfo**, and **acp_restrict** files to create diverse security profiles based on any combination of the *profile criteria* shown in Table A-52.

Table A-52. Profile Criteria

Criterion	Description
username	The user's userid.
group	The name of a group to which the user belongs, as defined in the **/etc/groups** or **acp_group** file; see *Creating User Groups* on page 15-447.
time	The day of the week and/or the time of day.
protocol	The connection protocol (e.g., PPP, CLI).
annex	The name or IP address of the Remote Annex on which the connection is made.
ports	The Remote Annex port number on which the connection is made.

Overview of Security Profile Criteria

Security profile criteria specify the connection conditions that must be met in order for the Remote Annex to:

- Use a particular security regime for authentication (in **acp_regime**).
- Define the user environment that will be in effect upon login (in **acp_userinfo**).
- Permit or restrict access to hosts or host ports (in **acp_restrict**).

Together, the security regime, user environment, and host access restrictions define the security profile.

A profile criterion begins with one of the keywords listed in Table A-52 on page A-439. The keyword is followed by an = sign, which is followed by a value. No space is permitted before or after the = sign. The syntax is:

keyword=value

To enter more than one criterion, separate the criteria with semicolons(;). Keep the criteria on one line. Use the backslash (\) continuation symbol to extend the line beyond the right margin, if necessary. No spaces are allowed on either side of the semicolon or within the *value* field, with the exception of data for the **time** criterion (see *Time* on page 15-444). A particular keyword may appear only once in a line of criteria. An entire line of profile criteria is called a *profile criteria specification*.

For the **acp_regime** and **acp_restrict** files, the entry looks like this:

```
username=chris;time=”9:00am – 10:30pm Monday-Friday”;annex=annex03
```

For the **acp_userinfo** file, the entry looks like this:

```
user username=chris;time=”9:00am–10:30pm Monday-
Friday”;annex=annex03
```

A profile criteria specification cannot exceed 80 characters.

After the profile criteria specification, you specify the security measure(s) to be applied if the criteria are met. The following is an example from the **acp_userinfo** file:

```
username=chris;time=”9:00am-10:30pm”;annex=annex03
climask ppp end
end
```

When user *chris* connects to *annex03*, **erpcd** records all the conditions related to the connection – the userid and any group associations (as defined in the **acp_group** or **/etc/group** file), the Remote Annex and port that *chris* connects to, the time of connection, and the connection protocol – CLI, PPP, or SLIP. **erpcd** saves these connection conditions for comparison with profile criteria specifications in the **acp_regime**, **acp_userinfo**, and **acp_restrict** files.

All of the profile criteria in a specification must be met in order for **erpcd** to consider that the specification *matches* the connection conditions recorded. The specification for Chris is considered a match if he gives a user name of *chris* and connects to *annex03* between 9:00 A.M. and 10:30 P.M. If all of these criteria are met, user *chris* is prevented (via the *climask* entry) from issuing the **ppp** command while he is logged into *annex03*.

Note that, in the example cited, the profile criteria specification replaces the **user** field in **acp_userinfo**. The **user** field can be specified instead, for compatibility with earlier Remote Annex releases. For more information, see *Creating the acp_userinfo File* on page 15-454.

One Match per File

You can enter an unlimited number of profile criteria specifications in each of the **acp_regime** and **acp_userinfo** files. However, for any single set of connection conditions, **erpcd** uses only the first matching specification it finds in each file. Consequently, the placement of profile criteria specifications is important. For example, suppose that user *chris* belongs to a group named *engineering* and that the first line in **acp_regime** specifies that the *engineering* group should be authenticated via Kerberos, while the second line specifies that user *chris* should be authenticated by SecurID. The result is that *chris* is authenticated by Kerberos, since a match for the group entry is found first.

The first-match algorithm is also true for **acp_restrict** entries that apply to CLI (**telnet** and **rlogin**) connections. However, **acp_restrict** entries for PPP and SLIP are treated differently (see *Limiting Access to Hosts via acp_restrict* on page 15-472).

The Resulting Security Profile

Once **erpcd** has found all the matching profile criteria in **acp_regime**, **acp_userinfo**, and **acp_restrict** (using the one-match-per file rule where appropriate) for a given set of connection conditions, the result is a single security profile.

Profile Criteria Syntax

The following sections give the purpose and syntax for each of the different criteria you can include in a profile criteria specification. Additional information and examples are supplied in the sections on **acp_regime**, **acp_userinfo**, and **acp_restrict**.

Username and Group Criteria

The **username** criterion lets you control security based on the Remote Annex userid (the name the user specifies at login).

The **group** criterion lets you control security based on a user's membership in a group. You assign users to groups via either the **acp_group** file or the **/etc/group** file (see *Creating User Groups* on page 15-447). When a **group** profile criterion is specified, **erpcd** checks the **acp_group** file to find the users belonging to the group. If it cannot find an **acp_group** file, **erpcd** looks in the **/etc/group** file.

A wildcard (*) can be used to represent as many of the final characters in a **username** or **group** as can be removed and still leave the name unique. The following are examples of **username** and **group** criteria:

```
username=fritz
username=fri*
username=frank
username=fra*
group=finance
group=fi*
group=fun
```

The following designates all users:

username=*

Time

The **time** criterion lets you control security based on the day of the week, the date, and the time of day. The following are the four possible syntaxes:

```
time="day"
time="time1-time2 day1-day2"
time="time1 day1 – time2 day2"
time="time1 date1 – time2 date2"
```

Enclose the **time** criterion in quotation marks and specify the arguments as follows:

- For *day*, specify a weekday, e.g., Sunday or Monday. The time criterion will apply to that entire day. Weekday specifications observe minimum uniqueness and are not case sensitive.
- For *time1*, specify the beginning of a time range; for *time2*, specify the end of a time range. Use *hh*:*mm*[**am**|**pm**], where *hh* is the hour and *mm* is the minutes, as the format for each end of the range. If you do not include **am** or **pm**, the Remote Annex assumes you are using military (24-hour) notation. Both ends of a range must use the same type of notation – you cannot use military time for one part of a range and **am** or **pm** for the other.

  To indicate midnight, specify either 12:00am or 00:0. Specify noon as 12:00pm or 12:00. To indicate a 24-hour range, use either 00:00 – 23:59 or 12:00am – 11:59pm.

  Be sure to include the colon and minutes (**:***mm*) after the hour (*hh*). For example, 9:00am – 5:00pm is valid; 9am – 5pm is not.

  You cannot specify time ranges without also specifying either a range of days or a range of dates. A time range with only a single day or date is not permitted. For example, *time= 9:00am – 5:00pm Sunday* is invalid. The correct usage would be *9:00am – 5:00pm Sunday-Sunday.*

- For *date1*, specify the beginning of a month and day range, e.g., January15, February10; for *date2* specify the end of a month and day range. Month specifications observe minimum uniqueness and are not case sensitive.

The following are examples:

```
time=”9:00am-5:00pm Monday-Friday”
time=”9:00-22:00 Sunday”
time=”Wed”
time=”8:00AM Friday - 6:35PM Friday”
time=”10:30 Nov 30 - 21:30 Nov 31”
```

The **time** criterion applies to initial access by the user. For instance, in the first example above, the criterion is met if the user logs in at any time between 9:00 A.M. and 5:00 P.M. on Monday through Friday of any week in any month.

Annex and Ports

The **annex** and **ports** criteria let you control security based on the Remote Annex and Remote Annex port (or set of ports) that the user tries to access. You can use an asterisk (*) symbol as a wild card in place of a Remote Annex name or the host part of a Remote Annex IP address. The following are valid **annex** and **ports** specifications:

```
annex=annex03;ports=1-6;annex=annex04;ports=5
ports=1-10,25,30
annex=192.17.5.*
annex=*
```

The first example specifies ports *1* through *6* on *annex03* and port 5 on *annex04*. In the second example, *annex=** is implied. In the third and fourth examples, *ports=** is implied. The fourth example specifies all ports on all Remote Annexes, which is the default.

You cannot abbreviate the **ports** keyword.

Protocol

The **protocol** criterion lets you control security based on the protocol used to attempt access to a host or host port. Valid values are:

- **slip**
- **ppp**
- **cli** (for telnet and rlogin)

Specify a protocol criterion using the syntax:

protocol=*protocol_name*

To specify more than one protocol, enter them on different lines. For example, to specify both PPP and SLIP, enter:

```
protocol=ppp
protocol=slip
```

The default is any protocol.

Overview of Files Used to Define Security Profiles

Following are the files you use to define security profiles:

- **acp_group** or **/etc/group**. If you intend to assign different security profiles to different groups of users, you must first define the groups in the **acp_group** or **/etc/group** file.
- **acp_regime**. An initial **acp_regime** file is created by the Remote Annex **install** program. It is based on answers you supply to prompts from **install,** and it contains a single authentication scheme, such as **acp**, to be used for authenticating all Remote Annex users. It also contains the name of a password file, if the regime is **acp** or **kerberos**.

You can modify the initial **acp_regime** file so that different authentication schemes are used when particular criteria are met.

Do not confuse ACP, the Remote Annex's Access Control Protocol that controls all host-based security, with **acp**, one of several authentication systems (regimes) that can be used with ACP.

- **acp_userinfo**. This file allows you to configure login environments based on a single userid or one or more profile criteria. Configurable aspects of login environments include CLI commands to be executed at start-up, CLI commands not permitted during a login session, filter and route definitions, a CHAP secret token, and various AppleTalk session characteristics. You can also use **acp_userinfo** to deny login access.
- **acp_restrict**. You can use this file to restrict access to hosts or host ports based either on the Remote Annex that attempts the connection or on specified access criteria.

The following sections describe these files in detail.

Creating User Groups

One of the most useful aspects of customizing security is the ability to apply different access rights and restrictions to different groups of users. To associate individual users with one or more groups, you create entries in the **/etc/group** or **acp_group** file.

The **/etc/group** file already exists on most UNIX systems, so you may prefer to add entries to this file rather than entering them in **acp_group**, which you must create. Either file must reside in the install directory (default is **/usr/annex**) on the UNIX security host.

erpcd looks for **acp_group** first, only using **/etc/group** if it cannot find **acp_group**. To designate that **erpcd** should use **/etc/group** rather than **acp_group**, see *Changing the Expected File Names Used by ACP* on page 15-548.

The **acp_group** file must have the same format as the **/etc/group** file.

The following systems do not support the **acp_group** file: Ultrix, FreeBSD, and BSDI. On these systems, you must use the **/etc/group** file.

An **/etc/group** or **acp_group** file contains a one-line entry for each group. To retain compatibility with **/etc/group**, the **acp_group** file includes passwords and group ID fields, although ACP does not use them. Due to UNIX implementation, you must specify a value in each of these two fields, although what you choose to enter is arbitrary. The format for an ACP group entry in **acp_group** or **/etc/group** is:

groupname:*password*:*groupid*:*userlist*

The *groupname* field specifies the name of the group; the *userlist* field is a comma-separated list of user names belonging to the group. The number of characters allowed in *userlist* varies from one UNIX platform to another, depending on the limit each platform uses for its **/etc/group** file.

You enter *userlist* names in one continuous string. Depending on the length of this string, the list may appear on more than one line but is actually stored as a single line in the file. Spaces, tabs, and new-line characters are not permitted in *userlist*. Fields are separated by the colon (:) character.

Following are two sample **acp_group** (or **/etc/group**) entries:

```
accounting:x:0:kim,herbert,sam,louise,bill
engineering:x:0:dilbert,jim,sharon,scott,john,liza,
carrie,edna,dena,caroline,marsha,sue,don,phil,eric,
dan,fritz,jeremiah,amy
```

The *x* and *0* (zero) in the previous examples are placeholders for the values that UNIX requires but ACP ignores. (On most UNIX systems, the *groupid* must be a numerical value.)

Configuring the acp_regime File

The initial security regime that the Remote Annex uses to authenticate all users is defined in the **acp_regime** file. This file is created the first time the network administrator runs the Remote Annex install program. The program prompts for a security regime and, if the regime requires it, a password file name. The regimes from which you can choose are **acp**, **securid**, **safeword**, **kerberos**, **native**, and **none** (see Table A-53 on page A-450). Password files are required for **acp** and **kerberos**; the defaults are **acp_passwd** and **/temp/tkt_erpcd_**. Both the **acp_regime** file and the password files (if any) must be stored in the installation directory, which defaults to **/usr/annex**.

The **acp_regime** file created by **install** has the following format:

:*initial_regime*[**:***initial_password_file*]

Once the **acp_regime** file has been created by the **install** program, you can modify the file to specify more than one regime and to include profile criteria that determine the conditions under which different regimes are used. The syntax for an **acp_regime** entry is:

[*profile_criteria*]**:***regime*[**:***password_filename*]

Each field in the entry must be separated by the colon (:) character; a space may follow but not precede the colon. The syntax for *profile_criteria* is:

keyword=*value*[**;***keyword*=*value*;...]

If profile criteria are omitted, the specified regime applies to any user logging in under any circumstances. Profile criteria are explained in *Overview of Security Profile Criteria* on page 15-440. Valid regimes are explained in Table A-53 on page A-450.

The *password_filename* field is valid only for the **acp** and **kerberos** regimes. If you specify the **acp** or **kerberos** regime but supply no *password_filename*, the default is used (see Table A-53 on page A-450). If the file is not found, an error message is logged and access is denied.

Table A-53. Authentication Regimes

Regime	Description
acp	ACP authentication, using the password file you specify. Default is the **acp_passwd** file.
safeword	SafeWord authentication.
kerberos	Kerberos authentication, using the ticket-directory prefix you specify. Default is **/temp/tkt_erpcd_**.
native	Authentication via the native operating system of the security server. For UNIX, native means the **/etc/passwd** file is used for authentication.
none	No authentication is performed; the user is unconditionally authenticated.
securid	SecurID authentication.

The following is a sample **acp_regime** file:

```
username=jack;time=”9:00am-10:00pm”:securid
group=finance:acp:special_acppw
:acp
```

Given this sample, **erpcd** uses SecurID to authenticate user *jack* if he logs in between 9:00 A.M and 10:00 P.M. (Since no day or date is given, this specification applies to all days and dates.)

Next, **erpcd** looks in the **acp_group** (or **/etc/group**, if **acp_group** does not exist) file to find the members of the group named *finance*. If one of these users tries to log in at any time on any day or date, **erpcd** attempts to authenticate that user via the **acp** regime, using the **special_acppw** password file (which must reside in the Remote Annex install directory). Even if user *jack* is defined in *finance*, if he logs in between 9:00 A.M. and 10: P.M., **erpcd** nevertheless tries to authenticate him via SecurID, since the profile criteria specification that begins with **username** is matched first.

Finally, any users whose login characteristics do not match the first two profile criteria specifications are authenticated via ACP, using the default password file, **acp_passwd**.

Creating User Password Files

Password Files for the acp regime

If the security regime defined is **acp**, **erpcd** prompts the user for a user name and password. The Remote Annex validates this information by instructing the security server to compare these entries against entries in the password file specified in **acp_regime**. If no password file is specified, ACP uses **acp_passwd**, which must reside in the install directory (default is **/usr/annex**). In either case, if a match is found, the user is granted access; otherwise, the user is denied access.

A typical session looks like this:

```
Annex Command Line Interpreter * Copyright 1988, 1995 Xylogics, Inc.
Checking authorization, Please wait...
Annex username: kate
Annex password:
Permission granted
annex:
```

The **acp_passwd** file uses the same format as the **/etc/passwd** file. The easiest way to create this password file is to copy the **/etc/passwd** file to **acp_passwd**. One advantage to creating the **acp_passwd** file this way is that you can merge **/etc/passwd** files from different hosts into one file on the security server, thus allowing you to create a network-wide password file.

If you are using a System V.4 or V.5 host, use the **/etc/shadow** file rather than the **/etc/passwd** file.

Not all password files work with ACP. For example, you could not merge SCO UNIX password files into the **acp_passwd** file.

Non-superusers can change their passwords only if the *username* in the **acp_passwd** file matches the *username* in the **/etc/passwd** (or **/etc/shadow**) file on the host.

Port Passwords for the acp regime

To have ACP prompt for a port password along with the user name and password, create an entry in the **acp_passwd** file as follows:

```
<Annex IP address>.<port_number>::0:0:<test>::
```

In the following example, the **acp_passwd** entry for the Remote Annex on port 1 called *Ollie*, with the IP address of 132.245.33.11, is:

```
132.245.33.11.1::0:0:Ollie Dialin modem port password::
```

After creating this entry, use the **ch_passwd** command to enter the port password:

```
% ch_passwd 132.245.33.11.1
New password: <password>
```

This port password is independent of the port parameter **port_password**. The port parameter is used only for local security.

The ACP prompts appear as follows:

```
Annex username:
Annex password:
Port password:
```

Password File for the Kerberos regime

If **kerberos** is defined in **acp_regime**, **erpcd** validates the user name and password by comparing them to entries in the password file specified in **acp_regime**. If no password file is specified, **erpcd** looks for /**temp/ tkt_erpcd_** in the install directory (default is **/usr/annex**). If **erpcd** does not find a match in that file, the user is denied access to the Remote Annex. For more information, see *Using Kerberos Authentication* on page 15-522.

Password Files for Other regimes

For information on passwords used with third-party systems other than Kerberos, see the following sections:

- *Using the SecurID Card* on page 15-524.
- *Using SafeWord AS Security* on page 15-534.

Password Histories and Blacklisting

You can enhance security for passwords by configuring the Remote Annex to record password histories and to blacklist users who have a configurable number of failed login attempts (for more details, ee *Enhancing Password Security* on page 15-486).

Creating the acp_userinfo File

The **acp_userinfo** file resides in the install directory and is maintained by the network administrator. The file primarily defines aspects of the user login environment. This environment can be defined on the basis of profile criteria and/or a userid (user name).

The information from **acp_userinfo** is loaded into the **erpcd** internal database. To update the database, send a USR1 signal to **erpcd** (**kill –USR1** *pid*). When updating **acp_userinfo**, it is a good idea to check syntax using the **erpcd –u** *filename* command (see *erpcd* on page C-239).

To create entries in the **acp_userinfo** file, use the following format, which is referred to as a **user...end** block:

```
user {name| profile_criteria}
        entry
        :
        :
end
```

The syntax for *profile_criteria* is:

keyword=*value*[**;***keyword*=*value***;**...]

Entering profile criteria is described in detail in *Profile Criteria Syntax* on page 15-443.

If you use the *name* argument instead of *profile_criteria*, specify a valid userid. This argument is supported for compatibility with Release 10.1 and earlier releases but is treated as if it were the profile criterion **username**=*name*. In searching **acp_userinfo**, **erpcd** looks only for a first match, whether that match is a single userid or all the criteria in a profile criteria specification.

The following is an example:

```
user username=jill
      climask slip ppp end
end
user group=finance;time="8:00AM-6:00PM Monday-Wednesday"
      clicmd ppp end
user group=finance
      deny
end
```

In the above example, **user username=jill** can also be specified as **user jill.**

In this example, even if user *jill* is a member of the *finance* group and meets all of the criteria in that profile criteria specification, *jill* is not permitted to use **slip** or **ppp**, since the first match found is the userid *jill*. The remainder of the example specifies that the finance group is allowed to connect only if its members log in between 8:00 A.M. and 6:00 P.M. The CLI port they are connected to will be converted to **ppp** mode after the group members have been authenticated. At any other time, they are denied access.

You can specify the following *entry* options (the following subsections discuss these options in detail):

- accesscode
- clicmd
- climask
- deny
- filter
- route
- at_zone
- at_connect_time
- at_nve_filter
- at_passwd
- chap_secret
- dyndial_passwd

accesscode

The **accesscode** is a string for which the dial-back security user is prompted, if defined, to determine what type of dial-back operation should occur. For each dial-back user or for conditions that meet profile criteria, you can define zero, one, or several **accesscode** entries in the **acp_userinfo** file. Each **accesscode** entry can define the phone number, inbound and outbound modem pools, and the job name (see Table A-54). The syntax is:

accesscode *code*
    accesscode_entry
end

For information about proprietary IPX dial-back, which requires an access *code* of **ipx** for which the user is not prompted, see *Dial-back* on page 15-505.

Table A-54. Entries for accesscode in the acp_userinfo File

Entry	Description
code	The user is prompted for this code when logging onto a port defined for dial-back security (after the **accesscode** prompt).
accesscode_entry	A list of one or more of the **accesscode** entries: **phone_no**, **in_pool_name**, **out_pool_name**, **job**.
phone_no	Specifies the dial-back phone number with the format: **phone_no** *phone_no* *phone_no* is the phone number to be called. If this optional parameter is not specified, the user is prompted for the dial-back phone number. System administrators are encouraged to specify this entry to avoid compromising system security. Any characters accepted by the modem can be used here. Notice that the escape character (\) must precede each special character (* , # @ ! ; =).

(continued on next page)

Table A-54. Entries for accesscode in the acp_userinfo File (continued)

Entry	Description
in_pool_name	Specifies the name of the inbound modem pool with the format: **in_pool_name** *pool_name* *pool_name* is the name of an inbound modem pool. For the dial-back request to be initiated, the designated port must belong to the inbound pool.
job	Defines a specific CLI command. The default is the CLI. Each **accesscode** can have up to one job record, using the format: **job** *command* [*argument...*]**end** *command* is a CLI command name, e.g., **rlogin**. *argument...* is an option list of command-specific arguments, e.g., to remotely log the user *Morse* into the host *amos*, the job entry is: `job rlogin amos -1 Morse end`

The following example illustrates **accesscode** entries in the **acp_userinfo** file. When logging in, the user *cobb* is prompted for a user name, password, and **accesscode**. If *cobb* enters the information at each prompt, the Remote Annex determines whether or not *cobb's* access is via the inbound modem pool; if so, *cobb* receives an *annex* prompt. If *cobb* enters *promptphone*, the Remote Annex prompts for a phone number, drops the connection, and calls *cobb* back via the outbound modem pool.

```
user cobb
     at_passwd                  nedry
     at_zone                    xy-33net xy-55net end
     accesscode                 access
                      phone_no            9\,7654321
                      in_pool_name        inbound
                      out_pool_name       outbound
                      job                 rlogin calvin -1
cobb end
     end

     accesscode                           direct
                      in_pool_name        inbound
     end

     accesscode                 promptphone
                      in_pool_name        inbound
                      out_pool_name       outbound
     end
end

pool inbound
     ports                      10,29-31@hobbes
     ports                      1-3@simon
end

pool outbound
     ports                      11-15@hobbes
     ports                      1-3@briggs
end
```

clicmd

For a single user or for conditions that meet profile criteria, you can define one or more CLI commands and macros in the **acp_userinfo** file. These commands will be executed, in the order in which they are specified, if the profile criteria are met or the user name matches the userid supplied at login. If the Remote Annex detects an error in a command, **erpcd** stops sending commands, syslogs an error, and denies access to the user.

If there are no **clicmd** entries for a user, the user can issue all CLI commands.

Table A-55 describes the **clicmd** entry. The syntax can be either of the following:

clicmd *CLI_command* **end**
clicmd ... end

For **clicmd** to work, the **cli_secuirty** parameter must be set to **Y**.

Table A-55. Arguments for the clicmd Entry the acp_userinfo File

Argument	Description
CLI_command	Any user or superuser CLI command, or the name of a macro previously defined for the Remote Annex. Only one command or macro is allowed per **clicmd** entry, although a **user...end** block can contain multiple **clicmd** entries. After the final command in a **user...end** block executes, the CLI session ends. To continue the session, use the **clicmd** with the ellipses (**...**) argument explained next. (For descriptions of the CLI commands, see *Using the CLI Commands* on page C-121. For information on macros, see *Creating macro Entries in the Configuration File* on page A-360. The Remote Annex can store up to 128 macros.)
...	Specifies that the CLI session should not end when the last **clicmd** entry for a given user has been executed. Subsequent commands in the same **user...end** block are ignored. See examples below.

The **clicmd** entry is useful for configuring dedicated connections. In the following example, if user *kip* logs in at any time between 9:00 A.M. and 5:00 P.M. on any day, the Remote Annex executes the **ppp** command (after authenticating *kip* at the CLI level). The port to which *kip* is connected is thereby converted from CLI to PPP mode. When the PPP link goes down, *kip* is disconnected from the Remote Annex.

```
user username=kip;time=”9:00am – 5:00pm”
     clicmd ppp end
end
```

The next example does the same thing as the previous example, except that it does not disconnect *kip* when the PPP link terminates.

```
user username=kip;time=”9:00am – 5:00pm”
     clicmd ppp end
     clicmd ... end
end
```

climask

For a single user or for conditions that meet profile criteria, you can define a CLI command mask in the **acp_userinfo** file that limits which CLI commands the user(s) can execute (see *Masking CLI Commands* on page 15-554). Table A-56 describes the entry for **climask** in the **acp_userinfo** file. The syntax for adding the CLI command mask to a user profile is:

climask *command_list* **end**

Table A-56. Entry for climask in the acp_userinfo File

Entry	Description
command_list	A list of user-level CLI commands, separated by spaces, that are ***not*** available to the user. Valid values are **bg**, **call**, **fg**, **hangup**, **help**, **hosts**, **jobs**, **kill**, **netstat**, **rlogin**, **stats**, **stty**, **telnet**, **who**, **lock**, **su**, **slip**, **connect**, **services**, **ppp**, **arap**, **ipx**, and **none** (the default). The list of restricted command names is sent to the Remote Annex and the user is prevented from executing those CLI commands. Do not specify the same command as both a **clicmd** and a **climask** in a given **acp_userinfo** entry (for more details on CLI commands, see *Using the CLI Commands* on page C-121).

The **climask** entry allows minimum uniqueness for command names. If you specify an ambiguous command name, **climask** generates a warning but cannot prevent the user from issuing the command.

The following is an example of **climask**:

```
user username=sam;time=”9:00am-10:30pm”
     climask ppp arap end
end
```

If user *sam* logs into any Remote Annex between 9:00 A.M. and 10:30 P.M., he cannot issue the **ppp** or **arap** command. In all other situations, this particular **user...end** block is ignored. For example, if *sam* logs into a Remote Annex at 11:00 PM, the entry is ignored.

deny

For a single user or for conditions that meet profile criteria, you can deny access to the Remote Annex in the **acp_userinfo** file. If the profile criteria are met or the user name in the user entry matches the userid supplied at login, ACP refuses access to the Remote Annex. The syntax is:

deny

Table A-57. Entry for deny in the acp_userinfo File

Entry	Description
deny	A keyword indicating the user will be denied access to the Remote Annex. If used, **deny** should be the only entry in the **user...end** block. A message is logged in the ACP log file indicating why access is being denied. For CLI users, a message is displayed.

The following is an example of using **deny** in the **acp_userinfo** file:

```
user liza
     deny
end
user group=eng;time=”9:00am -10:30pm”
     clicmd ppp end
end
```

In this example, even if user *liza* is a member of the *eng* group, she is denied access, since **erpcd** finds the match with the userid first.

In the following example, no user is permitted to connect to any Remote Annex between 11:00 PM and 12:00 PM on any day of any week:

```
user time=”11:00pm – 12:00pm”
     deny
end
```

filter

For a single user or for conditions that meet profile criteria, you can define one or more IP filters in the **acp_userinfo** file. These filters can apply to PPP and/or SLIP packets. The syntax is:

filter *filter_definition* **end**

Table A-58. Entry for filter in the acp_userinfo File

Entry	Description
filter_definition	Defines a filter to apply to the port on which the user logs in. You can enter only one filter per line but multiple filters are allowed within one **user...end** block. Filters are applied in the order in which they are specified. Unlike the **filter** command, a filter specification in **acp_userinfo** does not start with the word **add** (since it is assumed that you are adding a filter) and does not contain the name of the login interface (since that is known).

You can also restrict the transmission and reception of SLIP and IP over PPP packets by using the **acp_restrict** file. Using **acp_restrict** for this purpose can be easier than using **acp_userinfo** because you do not have to enter actual filters in **acp_restrict**. Instead, you enter user-friendly statements from which filters are created for you.

Any filters you enter in **acp_userinfo** or arrange to have generated by **acp_restrict** will be combined with, and interpreted according to the algorithm used for, filters created by the superuser **filter** command. For more information, see *Filtering* on page A-249.

The following example creates a filter that discards any IP packets destined for address *132.245.4.33* – if transmission of such packets is attempted on the port from which user *sam* logs in.

```
user sam
     filter output include dst_address 132.245.4.33 discard end
end
```

Like all other **acp_userinfo** entries (except **deny**), the **filter** entry can be accompanied by other entries within the same **user...end** block. In the following example, not only is the above filter created, but a pre-defined macro named *special_setup* and the CLI command **ppp** are also executed for user *sam*.

```
user username=sam
     clicmd special_setup end
     filter output include dst_address 132.245.4.33 discard end
     clicmd ppp end
end
```

route

For a single user or for conditions that meet profile criteria, you can define one or more IP routes in the **acp_userinfo** file. You can enter only one route per line but multiple routes are allowed within one **user...end** block.

Routes in **acp_userinfo** are entered into the routing table when their interfaces become active, but they are not entered into the route cache. You cannot use a **route** entry in **acp_userinfo** to define a default route. The syntax for the **route** entry is:

route [**–h**] *dest mask gateway* [*metric*] **end**

Table A-59. Argument for route Entry in the acp_userinfo File

Entry	Description
–h	Defines the route as hardwired.
dest	Specifies the destination address of the route.
mask	Specifies the subnet mask for the destination address. You can enter the mask in dotted decimal notation, e.g., 255.255.255.0, or you can specify the mask by appending /*n* to the destination address, where *n* is the number of 1 bits in the mask, from left to right. For example, appending /24 specifies 255.255.255.0 as the subnet mask. (Table A-10 on page A-200 shows all possible values for the subnet bit count, with the resultant subnet masks.)
gateway	Specifies the IP address of the gateway (router) that is the next hop for the route. If you specify an asterisk (*) for gateway, the Remote Annex uses the port's remote address as the gateway.
metric	Specifies the number of hops to the destination. Values range from **1** through **15**; the default is **1**.

Typically, a **route** entry in **acp_userinfo** is used when a router attached to a small network dials into the Remote Annex but does not want to incur the overhead of running a routing protocol itself. Consider the configuration in Figure A-30.

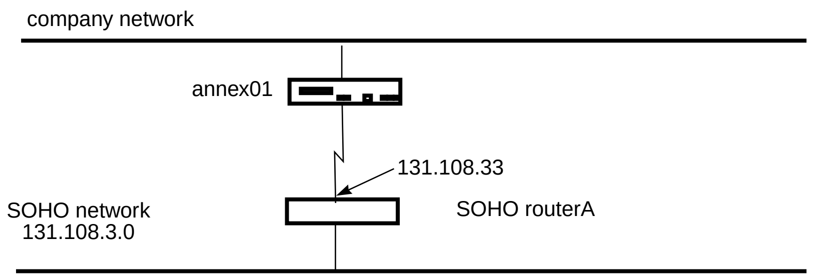


Figure A-30. Sample Configuration for a route Entry in acp_userinfo

Given the configuration in Figure A-30, the following example defines a route on *annex01* that will be used for the SOHO (Small Office/Home Office) router named *routerA*. This route allows packets to be sent back and forth between the company network and the network at IP address *131.108.3.0*. The destination address is *131.108.3.0*, using a subnet mask of *255.255.255.0*. The gateway is *131.254.33.1*, and the metric for the route is 1 (the default).

```
user username=routerA;annex=annex01
     route 131.108.3.0/24 131.254.33.1 1 end
end
```

at_zone

For a single user or for conditions that meet profile criteria, you can define AppleTalk zone list entries in the **acp_userinfo** file. This zone list consists of zone names (for more details, see *AppleTalk* on page A-301). Table A-60 lists the entries for **at_zone** in the **acp_userinfo** file. The syntax is:

at_zone *zone*... **end**

Table A-60. Entries for at_zone in the acp_userinfo File

Entry	Description
zone	A list of one or more ASCII character strings. You can have any number of zones specified in a zone list, subject to the following constraints: • A zone identifier cannot contain non-printable characters. • An individual zone identifier cannot exceed 32 characters in length. • The combined length of the entire zone list cannot exceed 524-*n* characters, where *n* is the number of zones in the list. • The reserved keyword **end** cannot appear as a zone argument. • A string containing a space must be enclosed in double quotation marks.

The following example illustrates **at_zone** entries in the **acp_userinfo** file. When logging in using ARA, user *cobb* is assigned to zones *xy-33net* and *xy-55net*.

```
user cobb
     at_zone xy-33net xy-55net end
end
```

The next example shows an **at_zone** entry that uses profile criteria. When logging in via ARA between the hours of 8:00 A.M. and 6:00 P.M, user *hobbes* is assigned to zones xy-11net and xy22-net.

```
user username=hobbes;time=“8:00am-6:00pm”
     at_zone xy-11net xy-22net end
end
```

at_connect_time

The **acp_userinfo** file can have an ARA connect timer defined; **at_connect_time** defines the maximum amount of time, in minutes, that an ARA connection can remain open. You can specify **at_connect_time** for a single user or for conditions that meet profile criteria. Table A-61 defines the argument for **at_connect_time** entries in the **acp_userinfo** file. The syntax is:

at_connect_time *time_value*

Table A-61. Entries for at_connect_time in the acp_userinfo File

Entry	Description
time_value	The format for this argument is <*minutes*>. For example: **at_connect_time** *120*

at_nve_filter

NVE filtering controls a remote access Apple user's view of network resources: when using *Chooser* to select resources, only the resource set defined for the user by the administrator will be visible. The administrator can specify the NVE filter on a per-user basis or for conditions that meet profile criteria. This feature complements the existing zone list, described above, by offering a higher level of control.

The administrator uses the **nve_filter** entry in the **acp_userinfo** file to specify a finite list of filter. Only one nve_filter entry per user or per profile criteria specification is permitted. The entry uses the format:

at_nve_filter [include|exclude] *tuple,tuple tuple...* end

Table A-62. Entries for at_nve_filter in the acp_userinfo File

Entry	Description
include| exclude	The **include** or **exclude** qualifier controls how filters are used: **include** filters allow only matching answers; **exclude** filters discard matching answers and allow non-matching answers. There is a 10-filters-per-user limit. The default is **include**.
tuple	A three-part string that identifies all network resources. The three parts of a *tuple* are: object, type, and zone. The format of a *tuple* is: **object:type@zone** with asterisks as wild cards. Any *, @, or : used as an NVE character within a *tuple* must be preceded by the Escape (\) character. Characters in a *tuple* are case-insensitive. Each field of an entity can be 32 characters long.

Following are sample **acp_userinfo** entries, including **nve_filter** information, for two users. User *frick* is allowed access only to the resources of her office Macintosh named *Frick CPU*. User *frack* cannot access *frick*'s machine, nor is she allowed access to any sales resources.

```
user frick
     at_passwd klot
     at_nve_filter include Frick\CPU:*@eng end
end

user frack
     at_passwd curly
     at_nve_filter exclude Frick*:*@ *:*@sales end
end
```

Like all other **acp_userinfo** entries, **nve_filter** information is syntax-checked by erpcd. Any errors cause the entire filter to be discarded, and an error message is generated.

This method of limiting NBP traffic is not secure, and can be circumvented by a person willing to write code to probe the network without using NBP. Also, this feature has no local Remote Annex security equivalent.

at_passwd

Each registered AppleTalk user (as opposed to a guest) must have a password defined in the **acp_userinfo** file. The definition can be for a single user or for conditions that meet profile criteria (e.g., membership in a group). Table A-63 defines the argument for **at_passwd** entries in the **acp_userinfo** file. The syntax is:

at_passwd *string*

Table A-63. Entries for at_passwd in the acp_userinfo File

Entry	Description
string	A string of up to nine alphanumeric characters (the un-encrypted password). Punctuation marks are permitted; spaces and hex values must be preceded by a backslash (\).

The following example illustrates an **at_passwd** entry in the **acp_userinfo** file:

```
#Set up the user entry
#

user cobb
     at_passwd ned\ ry
end
```

A guest entry in the **acp_userinfo** file looks like this:

```
#Set up a guest user entry that allows guests to connect
#for 1 hr.and hides our file servers

user <Guest>

     at_connect_time 60:00
     at_nve_filter exclude

        *:AFPServer@*

     end

end
```

The *Guest* entry is case-sensitive. If it is entered incorrectly, guests can login with no restrictions because the **at_guest** parameter for this port is set to **Y**.

chap_secret

A secret token that enables CHAP authentication for PPP is defined in **chap_secret** entries in the **acp_userinfo** file. The token can be defined for a single user or for conditions that meet profile criteria (e.g., membership in a group). Table A-64 defines the argument for **chap_secret** entries. The syntax is:

chap_secret *secret_token*

Table A-64. Entries for chap_secret in the acp_userinfo File

Entry	Description
secret_token	A string from 1 to 32 bytes long; 16 bytes is recommended due to the operation of the MD5 encryption algorithm.

The following example illustrates a **chap_secret** entry in the **acp_userinfo** file:

```
user username=smith
     chap_secret achapsecrettoken
end
```

For more details on CHAP and secret tokens, see *Challenge-Handshake Protocol (CHAP)* on page 15-516.

Limiting Access to Hosts via acp_restrict

The ERPCD can restrict any CLI (i.e., **telnet** and **rlogin**), SLIP, or PPP request for IP access to a specific host or host-port combination. This security mechanism uses a host-resident file that lists the hosts and host ports to which access is restricted and specifies the Remote Annex or the profile criteria to which the restrictions apply. By default, there are no host or host-port restrictions.

Host access security for CLI ports is enabled by setting the port parameter **connect_security** to **Y**.

Hosts or ports not listed in **acp_restrict** are considered unrestricted.

When a user issues a connection command or a SLIP or PPP link becomes active, the Remote Annex, using **erpcd**, checks a restrict file for permission to connect to that host. **erpcd** expects the restrict file to be **acp_restrict** (located in the installation directory), which is an ASCII file that you create]with any text editor. Table A-65 describes the arguments in each entry. The entry format is:

annex | *profile_criteria*: *restricted host* [[*ports*]] [,*restricted host* [[*ports*]], ...]

annex | *profile_criteria*~ *unrestricted host* [[*ports*]] [, *unrestricted host* [[*ports*]],...]

Table A-65. Arguments in the acp_restrict File Entries

Argument	Description
annex	The name or IP address of the Remote Annex initiating the access. This argument is supported for backward compatibility with Release 10.1 and earlier releases but is treated as if it were the profile criterion **annex=***annex* (*profile_criteria* are described next).
profile_criteria	One or more *keyword*=*value* pairs, separated by semicolons (;), specifying the conditions under which the specified hosts will be restricted or unrestricted. For information on entering *profile_criteria*, see *Profile Criteria Syntax* on page 15-443.
: (colon)	Indicates that the hosts listed in the same entry are restricted. White space may follow but not precede the colon.
~ (tilde)	Indicates that the hosts are unrestricted. White space may follow but not precede the tilde. For PPP and SLIP connections, hosts specified as unrestricted imply that all other hosts are restricted. (For a CLI connection, hosts specified as unrestricted have no implications for other hosts.)
restricted host	The name or IP address of a restricted host (including Remote Annexes). The list of restricted hosts is separated by commas; no white space is allowed. An asterisk (*) can be used as a wild card in place of a host name or the host part of an IP address.

(continued on next page)

Table A-65. Arguments in the acp_restrict File Entries (continued)

Argument	Description
unrestricted host	The name or IP address of an unrestricted host (including Remote Annexes). The list of unrestricted hosts is separated by commas; no white space is allowed. An asterisk (*) can be used as a wild card in place of a host name or the host part of an IP address.
[*ports***]**	One or more TCP or UDP ports on *restricted host* or *unrestricted host*. To specify multiple ports, separate them with commas or specify them as a range separated by a hyphen (–). Enclose the port(s) in square brackets **([ ])**. White space and wild cards are not allowed. The default is any TCP or UDP port.

Following are two restricted-host entries:

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

In the previous example, the first entry prevents SLIP, PPP, snd CLI connections from *annex01* to any port on *hosta*, *hostb*, *hostf*, or the host at IP address 132.245.6.23. The second entry prevents SLIP, PPP, and CLI connections from *annex02* to any port on *hostc*, *hostf*, *hosth*, the host at IP address 132.245.6.15, the host at IP address 132.245.6.23, and *annex01*.

In the next example, which shows the use of profile criteria, user *carl* is blocked from using **telnet** or **rlogin** to access hosts *atlas* and *steam*:

```
username=carl;protocol=cli:atlas,steam
```

For profile criteria entries in which the only protocol specified is **cli**, as in the previous example, **erpcd** searches the file in sequential order and uses only the first entry whose profile criteria are met. For these types of entries, order of placement in the file is important. If permission is granted to a CLI connection 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.

For profile criteria specifications that explicitly specify **slip** or **ppp** (or implicitly specify them, by not specifying *any* protocol), filters are automatically generated to restrict SLIP and/or PPP connections if either protocol becomes active. Consider the following example:

```
username=*;protocol=slip: finance
```

In this example, all SLIP users on all Remote Annexes are denied access to host *finance* but are allowed access to all other hosts and host ports.

Given an address of 132.245.11.4 for host *finance*, the filters generated to effect these restrictions are:

```
in include address_pair 132.245.11.4 * discard
out include address_pair 132.245.11.4 * discard
```

In the next example, the members of the group *mail_only* who connect using the PPP or SLIP protocol (as opposed to the CLI) may access the SMTP port (25) on host *mailhub* and the DNS server port (53) on the host *dns_srv*, but they cannot access anything else.

```
group=mail_only;protocol=slip~ mailhub[25], dns_srv[53]
group=mail_only;protocol=ppp~ mailhub[25], dns_srv[53]
```

To put these restrictions into effect, the Remote Annex would generate the following four filters, in which *132.245.33.1* is the address of *mailhub* and *132.245.33.2* is the address of *dns_srv.*

```
in exclude address_pair 132.245.33.1 * port_pair 25 * discard
out exclude address_pair 132.245.33.1 * port_pair 25 * discard
in exclude address_pair 132.245.33.2 * port_pair 53 * discard
out exclude address_pair 132.245.33.2 * port_pair 53 * discard
```

Filtering Restrictions

IP filtering can handle the following two cases:

- One or more hosts cannot be reached and all other hosts can.
- One or more hosts can be reached and all other hosts cannot.

However, IP filtering cannot handle the next two cases:

- A subset (e.g., a subnet or subnet group) of hosts can be reached, except for a few hosts in the subset, and all other hosts cannot be reached.
- A subset of hosts cannot be reached, except for a few hosts in the subset, and all other hosts can be reached.

For example, you cannot use **acp_restrict** to allow a user named *martha* to access all hosts on her home network (132.245.0.0), except for the finance machine at IP address 132.245.77.1, and also deny her access to hosts outside the 132.245.0.0 network. The **acp_restrict** entries for this would be:

```
username=martha: 132.245.77.1
username=martha~ 132.245.*
username=martha: *
```

If such an entry is found, a syslog message is generated and the user is denied access.

In addition, **acp_restrict** cannot create filters from host names containing wild cards, e,g., annex*.

Finally, filters apply to IP packets only; IPX and AppleTalk packets cannot be filtered.

Using include Files in the acp_userinfo File

To reduce the task of repeating several **job** and **climask** lines in the **acp_userinfo** file, you can create an **include** file. Nested **include** files are not allowed; the only commands allowed in the **include** file are **job** and **climask**. The syntax is:

include *filename*

Table A-66. Argument for the include File

Argument	Description
filename	The name of a file located in the same directory as **acp_userinfo**.

A log message is written to the ACP log file if the **acp_userinfo** file references an **include** file that could not be opened (*Host-based Security Logging* on page B-26 provides sample ACP log file entries).

Specifying Modem Pools Within the acp_userinfo File

A modem pool is a logical grouping of serial ports on one or more Remote Annexes. A minimum configuration includes at least one modem pool for dial-in and one modem pool for dial-out. A modem pool does not distinguish between dial-in and dial-out.

pool

The **pool** command defines a modem pool within the **acp_userinfo** file. Table A-67 lists the arguments for the **pool** command. The command syntax is:

pool *poolname*
        ports *port_set@annex*
end

Table A-67. Arguments for the pool Command

Argument	Description
poolname	The name of the modem pool.
ports	Each **ports** line defines a portion of the modem pool. Using multiple **ports** entries allows you to specify a modem pool that includes multiple *port_sets* on multiple Remote Annexes.

ports

The **ports** command defines the port(s) that belong to a given modem pool within the **acp_userinfo** file. Table A-68 lists the arguments for the **ports** command. The command syntax is:

ports *number@annex*

Table A-68. Arguments for the ports Command

Argument	Description
number@annex	A list of one or more port numbers.When specifying multiple port numbers, separate them with commas (meaning *and*) or hyphens (meaning *through*).

The following sample modem pool named *inbound* includes ports 1–16 and 24–32 on the Remote Annex called *titon*, and port 20 on the Remote Annex with the IP address *132.245.3.86*. Remote Annex entries are shown using symbolic and dotted quad notation.

```
pool inbound
     ports 1-16,24-32@titon
     ports 20@132.254.3.86
end
```

Dynamic Allocation of Network Addresses

Introduction to DHCP

The *Dynamic Host Configuration Protocol* (DHCP) enables dynamic IP addressing for remote PPP clients establishing a serial connection to a Remote Annex in a TCP/IP network. This eliminates the need to assign an IP address manually (and the subsequent need to reconfigure and reboot) each time that a host is added or moved to a new subnet location.

The Remote Annex acts as a DHCP “client-by-proxy”, requesting and accepting a dynamic IP address from a DHCP server on behalf of a dial-in client. The term *DHCP client* always refers to a Remote Annex acting as a DHCP client-by-proxy.

DHCP is enabled by setting the Remote Annex parameter **address_origin** (which replaces the previously existing parameter, **dialup_addresses**, for R13.2 and later) to **dhcp**, or by setting the Remote Address field in the **acp_dialup** file to **dhcp** (see *Determining Dial-up Addresses using the acp_dialup File* on page 15-482).

When DHCP is enabled, a DHCP client seeks to discover a DHCP server by requesting an IP address first from the DHCP server specified by **pref_dhcp1_addr**, then, if that server does not respond, from the DHCP server specified by **pref_dhcp2_addr**. The DHCP server checks the subnet of the requesting DHCP client, allocates an IP address from a pool of IP addresses made available for that subnet, and offers it to the requesting DHCP client. The DHCP client uses the allocated IP address for an interval of time called a “lease,” which is maintained for as long as the remote client connection is active, or until the DHCP client terminates the serial connection. When the lease expires, the DHCP client returns the address to the pool of dynamic addresses maintained by the DHCP server. The DHCP server can then reuse that IP address, allocating it to another DHCP client which requests an IP address.

Non-supported Features of DHCP

Some aspects of DHCP are not relevant to its use on a Remote Annex, specifically:

- A Remote Annex does not implement the BOOTP Relay function. A Remote Annex will not support host-based DHCP requests, e.g., a DHCP client operating on a PC, expecting the Remote Annex to relay host-generated DHCP protocol messages to the DHCP server.
- The DHCP client cannot be used to configure the dial-in host.
- DHCP clients do not use automatic address allocation, which assigns permanent IP addresses, nor do they retrieve statically configured IP addresses from the DHCP server.

Cautions

- If the DHCP client is invoked, but is unable to obtain an address from a DHCP server, it syslogs the condition “Client did not receive a DHCPOFFER”; the DHCP client cannot supply an address to the IPCP, and the remote connection is terminated.
- It is possible that the DHCP client will be unable to discover a DHCP server and obtain an IP address from it before the PPP connection establishment times out and terminates.

Creating the acp_dialup File

The **acp_dialup** file resides in the Remote Annex install directory. Any ACP dial-up address request that comes from the Remote Annex includes the Remote Annex address and port number, and an associated user name, which are used as keys in this file. Once the keys are matched, the corresponding dial-up addresses are returned to the caller on the Remote Annex. If no match is found, the Remote Annex uses the port’s **remote_address** and **local_address** (for more details, see *Determining Dial-up Addresses using the acp_dialup File* on page 15-482).

The **acp_dialup** file contains the following fields: *User*, *Annex*, *Remote Address*, and *Local Address* (optional). If a local address is not specified, the local address returned to the Remote Annex is that Remote Annex’s IP address. An example follows below:

```
#User   Annex/port_set        Remote address Local address
smith   4,5,6-9@100.30.200.39 100.30.200.45  100.30.200.46
green   *                     100.30.200.48
cody    3,7,20-25@jupiter     100.30.200.47
harris  mars                  100.30.200.55  100.30.200.40
frank   *                     dhcp
```

You can specify the Remote Annex by name, IP address, or wild card (*); the wild card means that any incoming address request with that user name will match. You can also specify a range of ports. The file format allows one entry per line; the Remote Annex ignores any data following the comment character (#); a newline character terminates an entry.

In the previous example:

- User *smith* can make a dial-up address request from Remote Annex 100.30.200.39 on ports 4, 5, and 6 through 9. The remote address is 100.30.200.45; the local address is 100.30.200.46.
- User *green* can make a dial-up address request from any port on any Remote Annex. The remote address is 100.30.200.48; the local address is the address of the Remote Annex from which the request originates.
- User *cody* can make a dial-up address request from the Remote Annex *jupiter* on ports 3, 7, and 20 through 25. The remote address is 100.30.200.47; the local address is *jupiter's* address.
- User *harris* can make a dial-up address request from any port on the Remote Annex *mars*. The remote address is 100.30.200.55; the local address 100.30.200.40.
- User *frank* will obtain a remote address from a DHCP server.

Determining Dial-up Addresses using the acp_dialup File

When the port parameter **address_origin** is set to **acp**, the *local* and *remote* field settings in the **acp_dialup** file supersede the values set in the **local_address** and **remote_address** port parameters.

In this case, the Remote Annex searches for the remote client's user name in the **acp_dialup** file. Remote Annex behavior at this point depends on whether or not the Remote Annex finds a matching user name in **acp_dialup**.

If the Remote Annex does find a matching user name in **acp_dialup**, it looks at the corresponding *local* and *remote* address fields.

- If both of these addresses are set in the **acp_dialup** file, the Remote Annex forces the use of these values over the settings in the **local_address** and **remote_address** port parameters.
- If the *local* address field is not set, but the *remote* address field is set, the Remote Annex forces the use of the *remote* address field setting for the remote address and forces the local address setting to be the Remote Annex's IP address.
- If the *remote* address field is set to **dhcp**, a remote address will be allocated dynamically by a DHCP server (see *Dynamic Allocation of Network Addresses* on page 15-479 for a complete description).

If the Remote Annex does not find a matching user name in the **acp_dialup** file, it looks at the **local_address** and **remote_address** port parameters.

- If the **local_address** and **remote_address** parameters are set, the Remote Annex uses these values for the local address and remote address.
- If the **local_address** and **remote_address** parameters are not set, the Remote Annex negotiates for both the local and remote address values with the remote PPP client. (If these conditions are true for a remote SLIP client, the connection is denied.)
- If the **local_address** parameter is set but the **remote_address** parameter is not set, the Remote Annex forces the use of the value in the **local_address** parameter and negotiates for the remote address value with the remote PPP client. (If these conditions are true for a remote SLIP client, the connection is denied.)

Using Dial-back Security

Dial-back security provides a level of security even if passwords are compromised. When a user dials into a port configured for dial-back security, the security host authenticates the user, drops the connection, and calls the user back at a pre-determined telephone number.

For maximum security, the system administrator should consider using dial-out only telephone lines with dial-back modems. In such a scheme the dial-in modems are used only to request the dial-back. Remote users using SLIP or PPP are authenticated in an interactive session before starting SLIP or PPP from the Remote Annex CLI.

In its default configuration, users are not required to have an entry defined in the **acp_userinfo** file. In this case, users are prompted for the user name and password only (no access code). After the authentication, a direct connection is established without dial-back. When the user does have an entry in the **acp_userinfo** file, the access code will be used to start the dial-back. This policy is useful in environments where security is not an issue but some users want to be called back. Only these (few) users require an entry in the **acp_userinfo** file.

System administrators can disable this policy and reconfigure **erpcd** so that all users are required to have a defined access code to access the Remote Annex (even if dial-back is not used by all users):

1. **Edit the file /usr/annex/src/erpcd/acp_policy.h:**

   Change the line “#define DEFAULT_NO_USERINFO” from a one to a zero.
2. **Rebuild** erpcd**.**
3. **Kill the current** erpcd **process.**
4. **Start the new** erpcd **process.**

The following sample pseudo code covers all cases for dial-back:

```
go through the username and password authentication
if (this Annex port number not included in any pool)
     {
     this is a direct connect
     }
else
     {
     prompt for access_code
     if (bad access code)
                      goto reject_request
     else if (port not member of inbound pool)
                      goto reject_request
     else if (outbound pool name not specified)
                      this is a direct connect
     else if (telephone number not specified)
                      prompt user for telephone number
     if (telephone number not specified)
                      goto reject_request
     else
                      dial back
     }
```

When using modems on dial-in ports with security enabled, turn off the *status message* and *result code* functions. Otherwise, the Remote Annex interprets the *status message* (i.e., CONNECT) as a user name trying to log in (for details on using modems with the Remote Annex, see *Modems* on page A-99).

For details on using a SLIP link, see *Serial Line Internet Protocol (SLIP)* on page A-137.

For details on using a PPP link, see *Point-to-point Protocol (PPP)* on page A-111.

Configuring the Remote Annex for Dial-back Security

The Remote Annex port parameter **type_of_modem** is a 16-byte string that specifies the modem type connected to the port(s) used for dial-back. The **type_of_modem** parameter indexes the modem description table in the **modem** section of the configuration file.

For more details on creating and using the configuration file, see *Parsing the Configuration File* on page A-345.

For more details on adding modem information to the configuration file, see *Creating modem Entries in the Configuration File* on page A-374.

For more details on using the configuration parameters, see *Configuration Parameters* on page C-33.

Enhancing Password Security

The following sections describe how to configure the Remote Annex to record password histories and blacklist users. It also explains how to view and manage the database in which password histories and blacklisting information is kept.

Overview of Password History and Aging

You can configure ACP to save the encrypted form of the previous passwords a user has set. This applies to passwords set by using the **ch_passwd** utility on the security host or by responding to a Remote Annex prompt when a password expires. These passwords are stored in the UNIX database **acp_dbm** on the security host, where they are keyed on user names. If a user tries to reset his or her password to one of the stored values, ACP will reject it and display an error message.

Benefits of Password Histories

The password history mechanism helps protect against off-line, “dictionary” attacks. In this kind of attack, a user obtains the encrypted **acp_passwd** (or /**etc/passwd**) file. The user then tries to crack the passwords by taking a dictionary of words, encrypting the words (using *salted* DES encryption) and comparing them to the encrypted passwords.

Benefits of Password Aging

The longer a password is in effect, the more time an attacker has to crack its encryption. Consequently, the password history feature is most effective when used in conjunction with password aging. If password aging is enabled:

- The user *must* change passwords when a predefined amount of time has elapsed. If the user never changes passwords, there is no password history to record.
- The user *cannot* change passwords until the predefined amount of time has elapsed. This prevents potential intruders from changing passwords in rapid succession in an attempt to cycle the old passwords out of the password history and use them again.

Password aging is enabled through the use of a **shadow** file in conjunction with a **passwd** file. By default, **erpcd** uses the **acp_passwd** file alone, so password aging is initially disabled. When only the **passwd** file is used (a Berkeley standard), that file contains both the user names (UIDs) and the encrypted passwords. The **passwd/shadow** form (used with UNIX System-V) contains an *x* in place of a password in the **passwd** file and saves the encrypted passwords in a separate file called **shadow**.

If your UNIX is based on System V and you want to use the password history feature, choose the **passwd/shadow** scheme. Use the *convert* program, located in the **erpcd** directory, to change the integrated **passwd** form to the **passwd/shadow** form (and vice-versa).

If your UNIX is based on a Berkeley BSD system, password history is disabled by default. To enable it, change the value of the *STORED_PASS #define* statement in **acp_policy.h**, as described in the following section.

Enabling and Configuring Password Histories

The following explains how to turn on the password history feature and (optionally) how to enable aging via shadow files.

1. **Use** na **or** admin **to make sure that the** enable_security **parameter is set to** Y **for the Remote Annex(es) you are configuring.**
2. **Use** na **or** admin **to make sure that you have defined a security host for the Remote Annex(es) you are configuring. (See *Configuring the Security Server* on page 15-434.)**
3. **Log into the security host as** root**.**
4. **Go** (cd) **to the installation directory (typically** /usr/annex**).**
5. cd **to the** src/erpcd **directory, which is within the installation directory.**
6. **In the** erpcd **directory, use a text editor to modify the** acp_policy.h **file.**
   - If you are using a shadow file, uncomment the following line in **acp_policy.h**:

     ```
     /* #define USESHADOW 1 */
     ```

     To uncomment the previous line, delete the slashes and asterisks at the beginning and end of the line, so that the line is as follows:

     ```
     #define USESHADOW 1
     ```

- The next **acp_policy.h** variable after USESHADOW is STORED_PASS, which is already uncommented. It defines the number of passwords that will be stored to prevent them from being re-used. The variable is initialized to 6 for **passwd**/**shadow** files and 0 for **passwd** files alone. A value of 0 disables password history.

  ```
  #ifdef USESHADOW
  #define STORED_PASS 6
  #else
  #define STORED_PASS 0
  #endif
  ```

  If you are using a **shadow** file and want to change the number of passwords stored from 6 to some other value, do so. The maximum is 12.

  If you are using a **passwd** file alone and you want to enable password history, change the value of the second STORED_PASS from 0 to a number from 1 through 12.

  Specifying a non-zero value for either of the above STORED_PASS variables turns on the recording of password histories in **acp_dbm**.

- The final variable related to password history is MAX_STORED_PASS. It defines the absolute maximum number of login failures that can occur before a user is blacklisted. It is best not to change this variable, which is set to 20. If you must change it, follow the instructions in **acp_policy.h**.

7. **If you plan to use only the password history feature and not blacklisting as well, follow the instructions in Steps 8 through 11, below. If you are also using blacklisting, wait to do this until you have configured both features.**

8. **From the** /usr/annex/src **directory, recompile** erpcd**:**

```
# cd /usr/annex/src
# make install
```

   This automatically rebuilds **erpcd** and any other files that need to be recompiled because of the changes you have made. In addition, the following message is displayed:

```
WARNING: If you have called “make install” yourself,
then in directory /usr/annex you will have to copy
erpcd.new to erpcd. Make sure the erpcd daemon is not
running when that is done.

If the installation script called “make install” then
the copy will be done for you.
```

9. **If** erpcd **is running on the host and the host is running Berkeley BSD UNIX, kill the existing** erpcd **process as follows. (Your process number will vary):**

```
# ps –ax | grep erpcd
25493 ? IW 0:00 ./erpcd
25797 p1 S 0:00 grep erpcd
# kill -9 25493
```

   To killl the process on a System V host, substitute the following for the first line above:

```
# ps –ef | grep erpcd
```

10. cd **to directory /usr/annex and copy** erpcd.new **to** erpcd**:**

```
# cd /usr/annex
# cp erpcd.new erpcd
```

11. **Now, restart** erpcd**.**

    ```
    # ./erpcd
    ```

After you complete Steps 1 through 11, the **acp_dbm** database is created automatically the first time a user changes a password via the **ch_passwd** utility. To list the users for which password histories exist, go to the security host's install directory (default is **/usr/annex**) and issue the **acp_dbm -l** command:

```
# cd /usr/annex
# acp_dbm -l
List of users currently present in the acp_dbm:
                     hobbes
                     fritz

#
```

In the previous example, password histories have been saved for users **hobbes** and **fritz**.

Overview of Blacklisting

A user account is considered under attack, and therefore blacklisted, when either (or both) of the following occurs:

- A configurable number (default is 5) of consecutive failed login attempts is exceeded. In other words, if you use the default, a user is blacklisted on the sixth consecutive failed login attempt.
- A configurable number (default is 10) of non-consecutive failed login attempts is exceeded within a configurable period of time (default is 26 weeks). If you use the defaults, a user is blacklisted when the eleventh failed login attempt occurs within a period of 26 weeks.

Blacklisting enhances security by limiting the number of passwords an on-line attacker can try before the user account is automatically disabled. At this point, no one can log in with the blacklisted user name, even if someone enters the “correct” password. However, the failed login message is the same before and after blacklisting, so the user does not know that the account has been disabled.

The system administrator is informed when blacklisting occurs. First, a record is created in the ACP log file indicating that the userid has been blacklisted. This record remains unless and until you delete it manually. Second, when you invoke the **acp_dbm** utility, it immediately displays a warning identifying any blacklisted users. See *Viewing and Managing the acp_dbm Database* on page 15-495.

The data necessary for blacklisting is kept in the **acp_dbm** database, keyed on the user name. If password history and blacklisting are configured, this database is created automatically the first time a user changes passwords or attempts to login and fails. The absence of an **acp_dbm** database indicates that no password histories exist and no failed login attempts have occurred.

Blacklisting makes the Remote Annex susceptible to denial-of-service attacks. To disable a user account, a saboteur need only make a few failed login attempts. In the extreme case, a saboteur who obtains a list of employee user names could create a shell script that would automatically disable all user login capabilities.

Configuring Blacklisting

You can configure blacklisting in one of two ways:

- By editing *#define* statements in the **acp_policy.h** file.
- By issuing the **erpcd** command with the **–b**, **–x**, and **–g** options.

The **erpcd** syntax is:

erpcd [**–b***max_con*] [[**–x***max_total*] [**–g***period*]]

Do not enter any space between an option (e.g., **–b**) and the value you specify with it (e.g., *max_con*).

The **erpcd** options override the **acp_policy.h** variables. Table A-69 describes the options and their **acp_policy.h** equivalents.

For information on how to edit the **acp_policy.h** file and put the modifications into effect (by rebuilding **erpcd** and the other files that have changed), see Steps 8 through 11 in *Enabling and Configuring Password Histories* on page 15-488.

Table A-69. erpcd Options and acp_policy.h Variables

erpcd Option	Equivalent acp_policy.h Variable	Description
–b*max_con*	MAX_BL_CON	The number of consecutive login failures a user is permitted before being blacklisted. Valid values are 0-8. A value of 0 enables blacklisting upon any login failure (not recommended). The default, as pre-set via MAX_BL_CON, is 5. If MAX_BL_CON is undefined and you do not specify **–b***max_con*, ACP never blacklists based on consecutive login failures.
–x*max_total*	MAX_BL_ NONCON	The number of non-consecutive login failures a user is permitted before being blacklisted. Valid values are 0-20. A value of 0 enables blacklisting upon any login failure (not recommended). The default, as pre-set by MAX_BL_NONCON, is 10. If MAX_BL_NONCON is undefined and you do not specify **–x***max_total*, ACP never blacklists based on consecutive login failures.
–g*period*	MAX_BL_ PERIOD	The time period, in weeks, over which *max_total* is applied. Login failures that occurred more than this number of weeks ago do not count toward blacklisting. Valid values are 0-52. The default, as pre-set via MAX_BL_PERIOD, is 26. If MAX_BL_PERIOD is undefined or is set to 0, MAX_BL_NONCON is effectively disabled.

Once you have configured and activated blacklisting, **erpcd** automatically creates the **acp_dbm** database the first time a user makes an unsuccessful login attempt. To monitor the blacklist status of one or more users, go to the directory (on the security host) that contains **`erpcd`** and use the **`acp_dbm`** utility, as described in the next section.

Viewing and Managing the acp_dbm Database

The **acp_dbm** utility lets you manage and display information about password histories and blacklisting from the **acp_dbm** database. To use this utility, you must log in with a userid of **root** or have superuser privileges. If neither is the case, **acp_dbm** immediately exits on invocation and displays the message:

```
You must have root privilege to run acp_dbm.
```

Execute the **acp_dbm** utility from the directory containing **erpcd**. Upon execution, **acp_dbm** immediately sends a warning message to standard output for each user on the blacklist. The message format is:

```
Warning: Annex user userid may be under attack; all logins
for this account have been disabled.
```

In this message, *userid* is the user name for the account that has been blacklisted.

The syntax for the **acp_dbm** utility is:

acp_dbm [**–s** *username*] [**–c** *username*] [**–d** *username*] [**–l**]

Table A-70 explains the options.

Table A-70. Options for the acp_dbm Utility

Option	Description

–s *username*	Sends information about *username* from the **acp_dbm** database to standard output. The output (after the initial warning message) shows the user name, the total number and type of failures, and the date and time of each failure. The following is an example:
	`User name: hobbes`
	`Total number of consecutive failed login attempts: 2`
	`     Login failure on Tue Dec 12 12:49:49 1995`
	`     Login failure on Mon Dec 11 11:25:10 1995`

–c *username*	Clears *username* from the blacklist and deletes all records of login failures for *username*. Does not clear the password history or any other information about *username* in the **acp_dbm** database. Before using this option, investigate the account thoroughly so that you are confident it is not under attack.
–d *username*	Deletes the user record from the **acp_dbm** database. Use this option, rather than **–c**, to delete the **acp_dbm** user account entirely. This option does not delete references to *username* in any other ACP files, such as **acp_userinfo** and **acp_passwd**. You must explicitly remove the user name from these files to delete the user completely.
–l	Lists all the user names contained in **acp_dbm**, including those with password histories.

Deleting the acp_dbm Database

The only way to delete the **acp_dbm** database is via the UNIX **rm** command.

Error Handling for Password Histories and Blacklisting

The following error conditions can occur:

- If **erpcd** cannot read or write to the **acp_dbm** database or detects incorrect protection, the event is syslogged at level LOG_CRIT and all users are denied access until **erpcd** can read and write to **acp_dbm**.

  If the wrong protection is detected, the syslogged message is:

  ```
  Security problem: Wrong protection (not 600) on acp_dbm database.
  ```

  If **erpcd** cannot read or write to **acp_dbm**, the message is:

  ```
  Cannot [read from | write to] acp_dbm database.
  ```

- If the **acp_dbm** utility fails to read or write the **acp_dbm** database, it generates the following message:

  ```
  acp_dbm: Error [reading from | writing to] acp_dbm database.
  ```

  If the utility detects the wrong protection, it generates the following message:

  ```
  acp_dbm: Wrong protection (not 600) on acp_dbm database.
  ```

- If the **ch_passwd** utility fails to read or write the **acp_dbm** database, **ch_passwd** generates the message:

  ```
  ch_passwd: Error [reading from | writing to] acp_dbm database.
  Notify System Administrator. Password change cancelled.
  ```

  If **ch_passwd** detects the wrong protection, it generates the message:

  ```
  ch_passwd: Wrong protection (not 600) on acp_dbm database.
  Notify System Administrator. Password change cancelled.
  ```

Using AppleTalk Security

The Remote Annex implementation of ARA provides three areas of security:

- ARA security
- Zone security
- NVE filtering
- Logging

ARA Security

The basic ARA security features are:

- **Username and password authentication**

  The Remote Annex authenticates the client using Apple's DES encryption algorithm. To define a user name and password for a registered (as opposed to guest) user, see *Creating the acp_userinfo File* on page 15-454.
- **Guest access**

  The Remote Annex allows anonymous access to the network. Restrictions can be applied to *guests* by setting up an ACP *guest* profile with limitations. For more details, see *at_zone* on page 15-466.
- **Connection timer**

  The connection timer is stored in the **acp_userinfo** file. For more details, see *Creating the acp_userinfo File* on page 15-454.

Zone Security

Every user can have a zone list assigned via remote ACP. If a list is not available via ACP, the Remote Annex provides all the zones it has learned from the network. If local security is used, use the per Remote Annex parameter **default_zone_list.** For more details, see *at_zone* on page 15-466.

NVE Filtering

NVE filtering controls a remote access Apple user's view of network resources: when using *Chooser* to select resources, only the resource set defined for the user by the administrator will be visible. The administrator can specify the NVE filter on a per-user basis. This feature complements the existing zone list, described above, by offering a higher level of control.

The **nve_filter** entry in the **acp_userinfo** file specifies a list of filters on a per-user basis. For detailed information on creating **nve_filter** entries, see *at_nve_filter* on page 15-468.

This method of limiting NBP traffic is not secure, and can be circumvented by a person willing to write some code to probe the network without using NBP. Also, this feature has no local Remote Annex security equivalent.

Logging

The Remote Annex logs activity and errors from the ARA session. The log is accessed via remote ACP (for more details, see *Logging User and Annex Events* on page B-26).

Using IPXCP Security

The Internet Packet Exchange Control Protocol (IPXCP) uses PPP security. For information on PPP security, see Using PPP Security on page 15-514. For information on other aspects of IPXCP, see *Internetwork Packet Exchange (IPX) Protocol* on page A-271.

Windows '95 IPXCP clients must make sure that SPAP security is not enabled on their PCs. SPAP is a proprietary Microsoft security mechanism not available to other systems, such as the Remote Annex.

Using Proprietary IPX Security

The proprietary Remote Annex implementation of IPX used by FastLink II clients provides the following security features:

- Authentication via passwords.
- Dial-back or charge-back services.
- Security logging.

The following sections describe how to configure security for the different IPX port types.

Security for ndp Ports

Configure security for **ndp** ports using the AMANAGER program described in the *Annex Manager for DOS Administrator's Guide for the PC*. The status of Remote Annex security has no effect on an **ndp** port, even if the Remote Annex **enable_security** parameter is set to **N**.

Security for ipx, auto_detect, and auto_adapt Ports

Initially, no security (other than a superuser password) is enabled for **ipx** or **auto_detect**/**auto_adapt** ports. The following sections describe how to configure security features for these ports. All features require that you set the Remote Annex **enable_security** parameter to **Y**.

If you set the port **mode** to **auto_detect** or **auto_adapt**, you must configure security for all protocols, in addition to configuring the security features described in the sections that follow. If you want to rely on CLI security, you must set the port **mode** to **cli**. To use IPX, the user then issues the CLI **ipx** command after being authenticated.

Configuring Password Authentication

To authenticate IPX users via passwords, use Remote Annex-based or UNIX host-based (ACP) security. Remote Annex-based security is the easier of the two to configure but allows only port password protection; you cannot use it for access control. UNIX host-based security can provide not only user-name and password authentication, but also dial-back/charge-back services (see *Dial-back* on page 15-505 and *Charge-back* on page 15-509) and security logging (see *Security Logging* on page 15-512).

Remote Annex-based Port Passwords

To set Remote Annex-based port passwords, set the port parameter **ipx_security** to **N** and set the **port_password** parameter (in addition to setting **enable_security** to **Y**). In the following example, passwords are set for ports 3 and 10 using **admin**:

```
annex: su
Password:
annex# admin
Annex administration Remote Annex Rx.x, 72 ports
admin: port 3,10
admin: set port ipx_security n
```

(continued on next page)

```
     You may need to reset the appropriate port, Annex
     subsystem or reboot the Annex for the changes to take
     effect.
admin: set port port_password wljc
     You may need to reset the appropriate port, Annex
     subsystem or reboot the Annex for the changes to take
     effect.
admin: reset 3,10
admin: show port port_password
port asy3:
     port_password: "<set>"
port asy10:
     port_password: "<set>"
```

When logging into the Remote Annex, an IPX user specifies a user name as well as a password, but the Remote Annex checks only the password.

UNIX-based Passwords

Instead of using Remote Annex-based authentication, you can use a UNIX host as a Remote Annex Access Control Protocol (ACP) server and store user names and passwords on it, or you can use a third-party security mechanism. To do this:

1. **Make sure the Remote Annex** enable_security **parameter is set to** Y**.**
2. **Define and configure a UNIX host as a security server, and, if you want, define and configure another one as a back-up. (See *Configuring the Security Server* on page 15-434.)**
3. **If you want the Remote Annex to broadcast for a security server when the defined servers do not respond, set the Remote Annex** security_broadcast **parameter to** Y**.**

4. **Set the** ipx_security **port parameter to** Y **for the** ipx **or** auto_detect/auto_adapt **ports you want to secure. The following example sets ports 3, 4, and 5 using** admin**:**

```
port 3-5
admin: set port ipx_security Y
       You may need to reset the appropriate port, Annex
       subsystem or reboot the Annex for the changes to
       take effect.
admin: reset 3-5
```

5. **Define the security regime(s) to be used and enter user names and passwords in the appropriate password file(s). For more information, see *Configuring the acp_regime File* on page 15-449.**

The following occurs when a proprietary IPX Fastlink II user logs into the Remote Annex from a PC client:

- The PC client passes the PC user name and password to the Remote Annex.
- Security software on the Remote Annex passes the user name and password to the UNIX security server. They are encrypted when sent, provided that the Remote Annex parameter **acp_key** has been set (see *acp_key* on page C-43).
  - If the UNIX server does not respond within a given amount of time (which you can configure), the Remote Annex calls the back-up security server, if defined.
  - If the back-up server fails to respond and **broadcast_security** is set to **Y**, the Remote Annex broadcasts to the network for a security server.
- If the username and password the PC user specifies match a username-password entry for the applicable security regime, FastLink II connects the PC to the Remote Annex and displays the DOS prompt.
- If no security server responds, or if the security regime does not define the user name and password, the Remote Annex denies the user access, and FastLink II displays an error message.

SecurID Passwords

The recommended way to use SecurID for proprietary IPX users is as follows:

1. **Make sure the Remote Annex** enable_security **parameter is set to** Y**.**
2. **Set a Remote Annex port to** cli**.**
3. **Have the user dial into this port from FastLink II terminal mode**.
4. **Have the user press the** Return **key.**
5. **Have the user enter the SecurID password when prompted for it.**
6. **Have the user issue the CLI** ipx **command.**

   This sets the port **mode** to **ipx** and connects the user to the Novell network.

If you set the port administratively to **ipx**, **npd**, **auto_detect** or **auto_adapt**, the user is prompted for a password before being connected to the Remote Annex. If SecurID times out, the user's password expires. When the user tries to reconnect using the expired password, FastLink II denies access to the Remote Annex and the user must enter a new password.

SecurID provides password authentication only. Use ACP for dial-back security or charge-back privileges.

The FastLink II auto-reconnect feature does not work with SecurID. Normally, by saving a disrupted session's state (including the user name and password), FastLinkII can dial the Remote Annex back and re-establish the user's session where it ended. However, with SecurID, the password expires before FastLink II can re-connect the user. The user receives an error message and must log in again.

The Remote Annex does not support the SecurID Network Loadable Module (NLM).

For information on configuring SecurID, see *Using the SecurID Card* on page 15-524.

SafeWord Authentication

To take advantage of all SafeWord features for authenticating IPX users:

1. **Make sure the Remote Annex** enable_security **parameter is set to** Y**.**
2. **Set a Remote Annex port to** cli mode**.**
3. **Have the user dial into this port from FastLink II terminal mode.**
4. **Have the user press** Return**.**

   The Remote Annex now prompts for the SafeWord authentication information you have configured.
5. **When authentication is complete, have the user issue the CLI** ipx **command. This sets the port mode to** ipx **and connects the user to the Novell network.**

IPX users connected to a port that is administratively set to **ipx**, **ndp**, **auto_detect**, or **auto_adapt** cannot be prompted for more than one password, nor can they receive a CHALLENGE.

For information on configuring SafeWord, see *Using SafeWord AS Security* on page 15-534.

Other ACP Security Mechanisms

In addition to password authentication, ACP also provides dial-back security, charge-back privileges, and security logging.

Dial-back

The dial-back feature increases network security by ensuring that particular users connect to the Remote Annex via authorized phone numbers. With dial-back, the Remote Annex automatically responds to a dial-in attempt by calling the user back at a predefined telephone number. You can assign up to nine dial-back numbers for each user. FastLink II provides a field in which the user specifies a number from the set of numbers you assign.

Annex Manager for DOS and FastLink II refer to dial-back as *call-back*.

The Remote Annex connects the user to one of a set of ports that you reserve for dial-in, and then calls the user back on one of a set of ports that you reserve for dial-back. By reserving different ports for dial-in and dial-back, you avoid tying up dial-in ports for long periods of time.

To configure dial-back:

1. **Make sure the Remote Annex** enable_security **parameter is set to** Y**.**
2. **For the incoming port:**

   a) Set the **mode** parameter to **ipx**, **auto_detect**, or **auto_adapt**.

   b) Set **ipx_security** to **Y**.

   c) Set the **type_of_modem** parameter to the modem type attached to the port (e.g., USR_144). This value must match the **type_of_modem** field in Remote Annex configuration file, which resides on the host from which your Remote Annex boots and is named (by default) **config.annex**.
3. **For the outgoing port (the port from which the user is dialed back):**

   a) Set the port mode parameter to **auto_adapt**, **adaptive**, or **slave**.

   b) Set the port **type_of_modem** parameter to the modem type attached to the port (e.g., USR_144). This value must match the **type_of_modem** field in Remote Annex configuration file, which resides on the host from which your Remote Annex boots and is named (by default) **config.annex**.

   c) Set the appropriate port configuration parameters for the outgoing modem, e.g., **set control_lines** to **both**.
4. **Enter the following in the** acp_userinfo **file. (If the file does not exist, create it; see *Creating the acp_userinfo File* on page 15-454.)**

   - A user name or profile criteria (see *Creating the acp_userinfo File* on page 15-454 and *Overview of Security Profile Criteria* on page 15-440).
   - An **accesscode** of **ipx**, which indicates the user has access to the IPX protocol on the Remote Annex.

- One or more **phone_no** entries specifying the telephone numbers at which the user can be dialed back. Include any additional digits that the user must dial, such as 9 (to obtain an outside line from a business phone) or 1 (for long-distance). You can use hyphens to separate logical parts of the number, e.g., to separate the 1 for long distance and the area code. You can also use one or more commas (,) to specify a modem pause, provided that you precede each comma with a backslash (\).

  When using FLSETUP to specify the dial-back number, the user must enter exactly what you specify in **acp_userinfo**, except for the backslash preceding a comma. Alternatively, the user can specify an integer (from 01 to 09) indicating the number's position in the **acp_userinfo** list. (An example is provided at the end of these instructions.)
- An **in_pool** name.
- An **out_pool** name.
- Pool entries defining the ports in the specified **in_pool** and **out_pool**. In order for the Remote Annex to initiate a dial-back request, the user must log into one of the ports defined in **in_pool**. The **out_pool** entry defines the ports on which the Remote Annex dials the user back.

  To keep a port free for dial-in, include it in **in_pool** but not in **out_pool**, or set its mode to **slave** (see Steps 2 and 3 for details on configuring these incoming and outgoing ports).

The following is an excerpt from an **acp_userinfo** file.

```
user zelda

     accesscode ipx
                      phone_no 9-1-212-555-0264
                      phone_no 9-555-0590
                      in_pool mktg_inpool
                      out_pool mktg_outpool
     end

end
pool mktg_inpool
     ports            29-32@stratplan
     ports            8-10@marcom
end
pool mktg_outpool
     ports            3-4@stratplan
     ports            5,6@marcom
end
```

In the preceding example:

- User *zelda* is configured so she can choose to be dialed back at either 9-1-212-555-0264 or 9-617-555-0590. Using the FLSETUP program, she can specify an actual phone number or an integer (from 01 to 09) indicating the number's position in the **acp_userinfo** list. For example, she can enter **9-555-0590** or she can enter **02** (see the documentation for the *FastLink II Client Pack)*.
- If *zelda* dials into port 29, 30, 31, or 32 on *stratplan*, *zelda* is dialed back from port 3 or 4 on *stratplan* or port 5 or 6 on *marcom*.
- If *zelda* dials into port 8, 9, or 10 on *marcom*, she is dialed back from port 3 or 4 on *stratplan* or port 5 or 6 on *marcom*.
- If *zelda* dials into any other port on any other Remote Annex, she is connected immediately rather than dialed back.

- If *zelda* is configured for dial-back but does not specify a call-back number via FastLink II, she is denied access to the Remote Annex.

Users need not be concerned about what ports the Remote Annex uses for dial-in or dial-back. Users need to know only the phone numbers they can dial into or the sequence numbers they can enter (which you must tell them before they use FastLink II).

If you subsequently add Remote Annexes to your network and want to protect them via dial-back, remember to add **ipx accesscode** and **pool** entries in the **acp_userinfo** file for them (and their ports).

If you change **acp_userinfo** while security is running, your changes do not take effect. To activate a new **acp_userinfo**, send a USR1 signal to **erpcd** to force it to update its copy of **acp_userinfo** using the UNIX command **kill**:

```
kill -USR1 pid
```

For more information, see *Re-compiling erpcd* on page 15-556.

Charge-back

Charge-back allows a user to dial into one or more Remote Annexes from any phone and be dialed back at any number. This is convenient for users, such as sales representatives, who want to dial in from more locations than can easily be configured for dial-back but who do not want sessions charged to the numbers from which they are dialing. Dial-back would limit the user to nine call-back numbers and require reconfiguration of the **acp_userinfo** file every time the user visits a new site. Charge-back eliminates both of these problems.

To configure charge-back, follow Steps 1–4 in the section *Dial-back* on page 15-505, with one change. Instead of specifying a telephone number in the **phone_no** entry, specify **charge_back**.

The following is an excerpt from an **acp_userinfo** file configured for charge-back.

```
user jeremiah

     accesscode ipx
                      phone_no charge_back
                      in_pool mktg_inpool
                      out_pool mktg_outpool
     end

end
pool mktg_inpool
     ports            29-32@george
     ports            8-10@pogo
end
pool mktg_outpool
     ports            3-4@george
     ports            5,6@pogo
end
```

User *jeremiah* can now dial into the Remote Annex named *george* or the one named *pogo* via any Remote Annex that uses this **acp_userinfo** file; *jeremiah* specifies the call-back number via FastLink II (see the *FastLink II Client Pack* documentation). After the Remote Annex authenticates *jeremiah* and his password, the following occurs:

- If *jeremiah* dials into port 29, 30, 31 or 32 on *george*, he is dialed back from port 3 or 4 on *george* or port 5 or 6 on *pogo*.
- If *jeremiah* dials into port 8, 9, or 10 on *pogo*, he is dialed back from port 3 or 4 on *george* or port 5 or 6 on *pogo*.
- If *jeremiah* dials into any other port on any other Remote Annex, he is connected immediately rather than dialed back.

Other possible outcomes are:

- If *jeremiah* is configured for charge-back and dials into an **in_pool** but does not specify a charge-back number via FastLink II, the Remote Annex accepts the call and connects *jeremiah* immediately.
- If *jeremiah* specifies a call-back number and is not configured for charge-back, the Remote Annex checks whether or not the call-back number matches a dial-back number defined in **acp_userinfo**.

  If the number matches a dial-back entry, *jeremiah* is called back at that number.

  If the number does not match a dial-back entry, but other numbers are defined for *jeremiah*, the Remote Annex denies him access.

  If no charge-back or dial-back numbers are defined for *jeremiah*, the Remote Annex connects him immediately.

  If you include both dial-back numbers and charge-back in the same **accesscode** entry, charge-back takes precedence.
- The user need not be concerned about which ports the Remote Annex uses for dial-in or charge-back.

When security is running, which is the case as soon as the **erpcd** daemon is started on the Unix host, **erpcd** makes an internal copy of **acp_userinfo**. If you change **acp_userinfo** while security is running, your changes do not take effect. To activate a new **acp_userinfo**, send a USR1 signal to **erpcd** to force it to update its copy of **acp_userinfo** using the UNIX command **kill** (for more details, see *Re-compiling erpcd* on page 15-556):

```
kill -USR1 pid
```

Security Logging

ACP host-based security generates audit trails of IPX user activity if you do the following:

1. **Make sure the Remote Annex** enable_security **parameter is set to** Y**.**
2. **Install ACP security software on one or more hosts (see *Setting Up a Security Server* on page 15-435).**
3. **Designate one of the hosts running ACP as a security host, and, if you wish, designate another as a back-up security host.**
4. **Set the** pref_secure1_host **and** pref_secure2_host **parameters to specify the preferred security hosts. Use the** set annex **command to set these parameters, e.g.,** set annex pref_secure1_host 0.0.0.0**.**

   The Remote Annex first queries the **pref_secure1_host** for user validation or permission connection requests. If a response is not received within the time in seconds defined in the **network_turnaround** parameter, the Remote Annex repeats the query several times. If the Remote Annex still does not receive a response, it queries the host defined in the **pref_secure2_host** parameter. If a response is not received from the second security host within a limited time, the Remote Annex broadcasts. If the broadcast fails, the Remote Annex denies the user’s request.
5. **Modify the** eservices **file (see *Configuring the Security Server* on page 15-434).**

When a user requests access to Remote Annex(es) configured for security logging, the Remote Annex calls the ACP security service on the security host. Each time this security service grants or denies a request to access a Remote Annex, the service enters a message in the ACP log file. This file is created by **erpcd** and is located in the install directory along with the other ACP files, such as **acp_passwd** (see *UNIX-based Passwords* on page 15-502).

If the Remote Annex cannot reach the security host, it calls the back-up host, if one is defined, and stores the message in the ACP log file on that host. If you examine this file on the primary security host and notice a gap in security entries, check the ACP log file on the back-up server. The primary server may have gone down temporarily; in this case, the events are logged to the secondary server.

If the Remote Annex parameter **security_broadcast** is set to **Y** (the default), and the Remote Annex cannot reach either the primary or back-up security host, it broadcasts for a security host on the network. If it finds one that is running **erpcd**, the Remote Annex stores any security messages in an ACP log file on that host, which could be anywhere on the network and therefore hard to locate. Set **security_broadcast** to **N** to avoid this problem.

The following is a sample ACP log file from a dial-back session:

```
Successful Dialback:

132.245.88.110:61060011:#05:940721:110630:ipx:login:\
CNA_FIXED,dialback request
132.245.88.110:61060011:#05:940721:110710:ipx:dial:\
CNA_FIXED, tel:568

No Outbound ports available  / port mode set wrong
132.245.88.171:4a0c0008:#02:940720:150441:ipx:login:\
CNA_FIXED,dialback request
132.245.88.171:4a0c0008:#02:940720:150614:ipx:dial:\
CNA_FIXED, ports busy


DSR low on outbound (dialback) modem:
132.245.88.171:4a0c0009:#02:940720:150753:ipx:login:\
CNA_FIXED,dialback request
132.245.88.171:4a0c0009:#02:940720:151259:ipx:dial:\
CNA_FIXED, srpc timed out
```

(continued on next page)

```
error from modem reset string received:
132.245.88.110:61060002:#05:940721:085212:ipx:login:\
CNA_FIXED,dialback request
132.245.88.110:61060002:#05:940721:085219:ipx:dial:\
CNA_FIXED, failed
```

To change the name and/or format of the ACP log file, see *Modifying the Supplied Security Application* on page 15-546.

The Remote Annex logs security events for **ndp**-mode ports as described in the *Annex Manager for DOS Administrator's Guide for the PC*.

Using PPP Security

The Remote Annex supports two authentication protocols for PPP:

- Password Authentication Protocol (PAP).
- Challenge-Handshake Protocol (CHAP).

Both of these protocols are run over the PPP link after the LCP negotiations are complete (for more details on using a PPP link, see *Point-to-point Protocol (PPP)* on page A-111.

Password Authentication Protocol (PAP)

PAP is a two-way handshake in which an ID/password pair are exchanged in clear text. Each half of the connection can require security.

If one side of the link agrees to use PAP, after the LCP negotiations are complete, that side will send a user name/password combination to its peer. Upon receipt, the peer authenticates that combination.

When the Remote Annex requests PAP and the peer ACKs the request, the Remote Annex handles the incoming PAP user name/password combination as follows:

- If the **enable_security** and **slip_ppp_security** parameters are set to **Y**, the Remote Annex first tries to authenticate the user name/password combination using ACP. ACP checks the regime file to determine the regime and password file to use (see *Configuring the acp_regime File* on page 15-449). If the ACP server is unavailable, the Remote Annex falls back to local security (i.e., it compares the remote end's user name/password against the port parameters **user_name** and **port_password**).
- If the **enable_security** parameter is set to **Y** and the **slip_ppp_security** parameter is set to **N**, the Remote Annex uses local security (i.e., it compares the remote end's user name/password against the port parameters **user_name** and **port_password**).
- If the user name/password combination is valid, the Remote Annex sends a *PAP Authenticate-ACK* message. If the combination is not valid, the Remote Annex sends a *PAP Authenticate-NAK* message.

When the Remote Annex agrees to PAP, it sends the PAP user name/password combination as follows:

- It uses the port parameter **ppp_username_remote** as the user name.
- It uses the port parameter **ppp_password_remote** as the password.
- If the user name/password combination is valid, the peer sends a *PAP Authenticate-ACK* message. If the combination is not valid, the peer sends a *PAP Authenticate-NAK* message.

Challenge-Handshake Protocol (CHAP)

CHAP is a three-way handshake that depends on a secret token. The secret token is known to both sides of the link. When the authenticator sends a *challenge* message to the peer side of the link, the peer responds with a one-way encrypted value. The authenticator then runs the same encryption and compares the result to the received value. If they match, the authenticator sends a *success* message; otherwise, it sends a *failure* message. Currently, the only encryption algorithm supported is MD5.

The secret token must be distributed to both sides of the link by an external mechanism.

ACP is used only when the Remote Annex is authenticating a peer.

CHAP does not use the **acp_regime** file.

The secret token is defined within an *entry* option called **chap_secret** in the **acp_userinfo** file (for more details, see *Creating the acp_userinfo File* on page 15-454 and *chap_secret* on page 15-471).

In the following example, user *smith*, when logging into a Remote Annex running CHAP, will have the secret token, *achapsecrettoken*, used in verifying the *response*. The mechanism of receiving a challenge, determining a secret based on the user, and sending the result back to the challenger, is analogous to the user name/password paradigm.

```
user smith
     chap_secret achapsecrettoken
end
```

If the **slip_ppp_security** parameter is set to **Y**, the Remote Annex uses ACP to acquire the secret token based on the name field in the *response*. The Remote Annex uses local security when ACP is unavailable and the **port_password** parameter is set; local security ignores the user name and checks the *response* against **port_password**. If the **port_password** parameter is not set, the link fails.

Receiving a CHAP Challenge

When the Remote Annex receives a *challenge*, it acquires the secret token (the **ppp_password_remote** parameter value) and generates a *response* message (the *name* field is set to the **ppp_username_remote** parameter value). The value in the *response* message is a result of running MD5 encryption on the secret token and the value in the *challenge* message. If the Remote Annex receives a *success* message, the link enters (or remains in) NCP negotiation; otherwise, the link is terminated.

The Remote Annex can receive any number of *challenges* at any time after LCP negotiations are complete.

The Remote Annex negotiates an authentication *challenge* from a peer only if the **ppp_password_remote** and **ppp_username_remote** parameters are set for this port.

CHAP does not use the **acp_regime** file.

Sending a CHAP Challenge

When the Remote Annex sends a *challenge*, it includes the **chap_auth_name** parameter value as the *name* field and a randomly generated number as the *value* field. If ACP is used, after receiving a *response*, the Remote Annex acquires the secret token based on the name in the *response* message (the Remote Annex uses the **port_password** parameter value as the secret token if local security is used).

If the result of running MD5 encryption on the secret token and the randomly generated number produce the same value as in the *response* value field, the Remote Annex sends a *success* message and the link enters NCP negotiation; otherwise, the Remote Annex sends a *failure* message and terminates the link.

The Remote Annex sends a *challenge* only if the **enable_security** parameter is set to **Y**, the **ppp_security_protocol** parameter is set to **chap**, and CHAP is ACKed during LCP. If the Remote Annex is ACKed for CHAP, it will seek only one valid *response*. Once the Remote Annex receives a valid *response*, it sends *challenges* at irregular intervals while the link is up.

The Remote Annex terminates a link if it cannot authenticate a *challenge*.

If the Remote Annex does not receive a *response* to a *challenge* within the allotted time-out, it re-issues the *challenge* for the defined number of retries.

ACP logging for CHAP includes good responses received, bad responses received, and peer refusal to do CHAP (for more details on security logging, as well as a sample log file, see *Host-based Security Logging* on page B-26).

Using the PPP Security Parameters

There are a variety of settings one can choose when configuring the Remote Annex for PPP security. Table A-71 lists the possible combinations of PPP security parameter settings and their effect on Remote Annex activity.

The following two statements are true for all cases listed in Table A-71. First, if a remote side of a link demands PAP, the Remote Annex uses **ppp_username_remote** and **ppp_password_remote** for the username and password. Second, if **ppp_username_remote** and **ppp_password_remote** are not set, the connection fails.

Table A-71. PPP Security Parameters and their Effect on Remote Annex Activity

If:	Then:

enable_security = N
ppp_security_protocol = n/a
slip_ppp_security = n/a
	Request no PPP security incoming.
	Do not log accesses in the ACP log file.

enable_security = Y
ppp_security_protocol = none
slip_ppp_security = Y
	Request no PPP security incoming.
	Log accesses in the ACP log file.

enable_security = Y
ppp_security_protocol = none
slip_ppp_security = N
	Request no PPP security incoming.
	Do not log accesses in the ACP log file.

enable_security = Y
ppp_security_protocol = pap
slip_ppp_security = Y
	Use ACP for incoming user name and password. Log accesses in the ACP log file.

enable_security = Y
ppp_security_protocol = pap
slip_ppp_security = N
	Use **port_password** for incoming password. Do not log accesses in the ACP log file.

enable_security = Y
ppp_security_protocol = chap
slip_ppp_security = Y
	Use ACP for incoming user name and secret token. Log accesses in the ACP log file.

enable_security = Y
ppp_security_protocol = chap
slip_ppp_security = N
	Use **port_password** for incoming secret token. Do not log accesses in the ACP log file.

enable_security = Y
ppp_security_protocol =
chap-pap
slip_ppp_security = Y
	Request CHAP in negotiation; if it is NAKed by peer, request PAP.
	For CHAP, use ACP for incoming user name and password. For PAP, use ACP for user name and secret token.
	Log accesses in the ACP log file.

(continued on next page)

Table A-71. PPP Security Parameters and their Effect on Remote Annex Activity (continued)

If:	Then:

enable_security = Y
ppp_security_protocol =
chap-pap
slip_ppp_security = N
	Request CHAP in negotiation; if it is NAKed by peer, request PAP.
	Use **port_password** for incoming password/secret token and ignore incoming user name.
	Do not log accesses in the ACP log file.

enable_security = Y
ppp_security_protocol = chap-pap
slip_ppp_security = Y
	Request CHAP in negotiation; if it is NAKed by peer, request PAP.
	If using CHAP, use ACP for incoming user name and password. If using PAP, use ACP for user name and secret token.
	Log accesses in the ACP log file.

enable_security = Y
ppp_security_protocol = chap-pap
slip_ppp_security = N
	Request CHAP in negotiation; if it is NAKed by peer, request PAP.
	Use **port_password** for incoming password/secret token and ignore incoming user name.
	Do not log accesses in the ACP log file.

Using Filters for Security

The Remote Annex implementation of filtering allows you to improve the security of an internal network by preventing potentially dangerous traffic from crossing it. For example, you might want to prevent an outside host from using the Network File System (NFS) protocol or the Trivial File System Protocol (TFTP) to access an internal network, since these protocols have no built-in security and can alter local data. Or, you might want to use filtering to prevent users on your internal network from accessing external hosts and services.

An effective way to provide this kind of protection is to pick one Remote Annex on the internal network to be the network's *chokepoint* or *firewall* through which all traffic to and from external networks must pass. Then, configure filters on that Remote Annex to block undesirable packets (see *add Subcommand Examples* on page A-264).

You can also use filtering to log (in the **syslog** file) traffic for security or network-management purposes (see *add Subcommand Examples* on page A-264). Finally, you can use filters to determine what constitutes traffic on a dial-out serial port.

Filters can apply to one particular physical interface on a Remote Annex or to all Remote Annex interfaces and can affect incoming or outgoing packets. An interface is a SLIP port named *asy**n***, where ***n*** is the port number, a PPP port named asy*n* (again, *n* is the port number), or the Ethernet port (*en0*). For more details on filtering, see *Filtering* on page A-249.

You need superuser privileges not only to configure the Remote Annex for filtering but also to create or modify filters.

Using Kerberos Authentication

The default ACP configuration authenticates a user by checking the user name and password against entries in the **acp_passwd** file. You can configure ACP to use Kerberos instead of the default authentication process.

When building the ACP/**erpcd** process, a Kerberos library routine (**libkrb.a**) is linked with the ACP code. ACP prompts the user for a user name and password. However, instead of validating the user name and password via the **acp_passwd** file, ACP opens a connection to the Kerberos server and passes the user name and password to the Kerberos library routine for authentication. The Kerberos library routine returns a ticket to ACP indicating whether or not the user is authenticated.

If the Kerberos server authenticates the user, it encrypts the ticket with the user's password before returning it to ACP. If the Kerberos server rejects the user, it returns an error code, and ACP refuses the login attempt. In either case, ACP calls a separate Kerberos routine to destroy the returned ticket after the validation process.

Enabling Kerberos Authentication

To enable Kerberos authentication, you must rebuild the **erpcd** process, and then use this process instead of the default version. To rebuild **erpcd**:

1. **Edit the** make.config file in the /annex_root/src directory **and look for the keyword CFG_STUBLINKING, at the bottom of the file. The line will look like this:**

   ```
   CFG_STUBLINKING = L. -lstubs
   ```

2. **Modify the line in Step 1 to include the libkrb.a file, as follows:**

   CFG_STUBLINKING = *Kerberos_lib_path*/libkrb.a

   For *Kerberos_lib_path*, specify the name of the directory containing the libkrb.a file. This file is located in the directory in which Kerberos was installed.

3. **Rebuild** erpcd **(see *Re-compiling erpcd* on page 15-556).**
4. **Install the new** erpcd **in the usual place (saving the old version as a back-up in case of problems).**
5. **Terminate the executing** erpcd **and start up the new version.**

If both the primary and secondary ACP servers are defined, it is important that both the primary and secondary ACP servers support Kerberos authentication for consistency.

Configuring the Remote Annex for Use with Kerberos Authentication

To configure the Remote Annex for use with Kerberos authentication, you must set the parameters as indicated in Table A-72.

Table A-72. Kerberos Parameter Settings

Parameter	Setting
enable_security	Yes
security_broadcast	No
port_server_security	Yes
vcli_security	Yes
vcli_password	<unset>
password	<unset>
cli_security	Yes (on each serial port)
port_password	<unset> (on each port)

Using the ACE/Server

The ACE/Server token is an access control security token which is used to positively identify users of computer systems and secure TCP/IP networks. Used in conjunction with the SecurID card hardware or software access control modules (ACMs), the ACE/Server token automatically generates a unique, unpredictable access code every 60 seconds. The ACE/Server, a daemon that interfaces with the user database, also allows the system administrator to monitor access by running reports of all attempted logins.

Supported ACE/ Server Releases

Remote Annex R4.1 offers support for ACE/Server Release 2.1.1 and 2.2.

ACE/Server is supported using ACP and is limited to those UNIX platforms for which the vendor provides client libraries.

Using the SecurID Card

To use the SecurID card feature, you must purchase the ACE/Server software from Security Dynamics. The ACE/Server software includes client software and the SecurID card. The ACE/Server system is designed to prevent any unauthorized access to your network.

SecurID Card Description

The SecurID card is a credit-card sized card containing a microprocessor and an LCD display. This card generates, at a designated interval, a one-time-only, unpredictable code on the LCD display. At the usual system prompt from your RA 6300, SecurID card users enter a passcode in order to access your protected system.

ACE/Server Authentication

The ACE/Server system provides a unique code, such as the user’s PIN number.

Each SecurID card has a unique serial number that identifies it to the ACE/Server.

Assigning a SecurID Card to a User

When you receive the ACE/Server software and SecurID cards, one of the cards is already assigned to the login name *adm* and is enabled for your system administrator. When you become *adm* and execute **sdadmin**, it determines that you have assumed that login name and uses it to find the **adm** card and your correct authority level. The **sdadmin** does not require a passcode entry. By using the one card with administrator authority, at least one person in your SecurID system has the authority to manage the ACE/Server system and all its databases, including changing any relevant information.

Clients

An ACE/Server UNIX Client is a TCP/IP machine connected via a network to the ACE/Server. Whenever a client sends a user-authentication request, the ACE/Server looks up the client's name. For this name to be found, all clients network addresses must be entered into the database, and all the network addresses must be known to the server via the **/etc/hosts** file or your NIS name server.

The SecurID Card User Interface

When a user tries to log into your system, the ACE/Server prompts for the user name and passcode. The user enters the PIN number followed by the current SecurID card code displayed on the SecurID card.

Access Types

The ACE/Server utility authenticates two access types:

- Port-to-port RA 6300
- Network-to-port

To authenticate SLIP, PPP, and IPX users:

Authenticate SLIP, PPP, and IPX

1. **Log into a CLI port.**
2. **Issue the CLI command** slip, ppp**, or** ipx**.**

   Or
3. **Log into** auto_detect **and** auto_adapt **ports.**
4. **Press** Return **to enter CLI mode, and then issue the** slip, ppp**, or** ipx **command.**

Generating PINs

When...	**The...**
a SecurID card is assigned to a user	card is set to New-PIN mode in the ACE/Server database.
a user attempts to log on for the first time to a network via a Remote Annex	user enters only the code on the SecurID card (if the PIN has been cleared).
SecurID requires a unique PIN	user must enter a new PIN (user- or system-generated).

The ACE/Server software provides three options related to generating a new PIN:

- CANNOT_CHOOSE_PIN
- MUST_CHOOSE_PIN
- USER_SELECTABLE

Before installing the ACE/Server software, you must determine which of the above options your site will use. The following is an overview of the available options. See the *ACE/Server Manual* for more information.

CANNOT_ CHOOSE_PIN

The new PIN is generated by the system and does not give the user the option to select a new PIN. The user is prompted to allow the system to generate and display the new PIN or exit and leave the SecurID card in New PIN mode.

MUST_ CHOOSE_PIN

The user must select a new PIN and is not given the option of having the system generate the new PIN. The user is prompted to enter a new PIN containing 4 to 8 alphanumeric characters or exit and leave the SecurID card in New PIN mode.

USER_ SELECTABLE

The user is given the option to select a PIN or have the system generate and display a new PIN. The user is prompted to enter a new PIN containing 4 to 8 alphanumeric characters or have the system generate a new PIN and display it or leave the SecurID card in New-PIN mode.

Installation

Copy Files to src/sdclient

During the Remote Annex software installation, you must copy the following library and files from your ACE/Server distribution media to the **src/sdclient** directory:

- sdclient.a library
- *.h files

The ACE/Server UNIX Client must be installed on each host running **erpcd**. For more detailed information, see the *ACE/Server Installation Guide*.

Each UNIX host running a SecurID-enabled **erpcd** must be enabled as a client in the ACE/Server. For more detailed information, see the *ACE/Server Administration Manual*.

Makefile Switches

Define Makefile Switches

Define a new set of switches in the **Makefile** by uncommenting the two lines that define ACE1_2 or ACE2_0 in **erpcd/Makefile** for ACE/Server V2.1.1 or V2.2. Also, comment out the flag (**PASSFLAG**) that causes the Remote Annex password prompt to appear:

Example

```
#SECURIDFLAG=-DSECURID_CARD -DACE2_0

#SECURIDFILES=../sdclient/sdiclient.a

...

PASSFLAG = -DPASS_SEC
```

to

```
SECURIDFLAG=-DSECURID_CARD -DACE2_0

SECURIDFILES=../sdclient/sdiclient.a

...

#PASSFLAG = -DPASS_SEC
```

To integrate SecurID into ACP, you must make changes in the **erpcd** utility. When you have made the necessary changes to the **Makefile**, rebuild the Remote Annex software. See *Re-compiling erpcd*, later in this chapter.

Define Makefile Switches for non-ANSI Standard Compiler

With a non-ANSI standard C compiler, uncomment the following lines in the **Makefile**:

```
#SECURIDCFILES=fflush.c

#SECURIDOFILES=fflush.o
```

to

```
SECURIDCFILES=fflush.c

SECURIDOFILES=fflush.o
```

New-PIN Mode

If the site allows a user to select a PIN, ACP displays the following text:

```
Enter your new PIN containing 4 to 8 digits,
     or
Press <Return> to generate a new PIN and display it,
     or
<Ctrl-D> <Return> to leave your card in New-Pin mode.
```

The minimum and maximum PIN lengths and the choice between digits only or alphanumeric characters is determined by the system administrator when installing the ACE/Server.

If the user enters a PIN, ACP prompts for the code's re-entry (the typed characters are not echoed back to the terminal). The re-entry prompt looks like this:

```
Please re-enter PIN:
```

If the user is not allowed to choose the PIN, the following text is displayed:

```
Press <Return> to generate a new PIN and display it,
     or
<Ctrl-D> <Return> to leave your card in New-Pin mode.
```

If the user presses **Return**, the terminal displays the assigned PIN. If the user incorrectly re-enters the PIN or chooses to leave the card in New-Pin mode, the login attempt is terminated.

Configuring the RA 6300 for Use with SecurID

To use the SecurID card, security must be enabled on the RA 6300:

1. **Set the following RA 6300 parameters to** Y**:**
   - **enable_security**
   - **vcli_security**
2. **Set the following RA 6300 port parameters to** Y **on the global port:**
   - **cli_security**
   - **port_server_security**
3. **Set the RA 6300 port parameter** ppp_security_protocol **to** none **on each port.**

   If **ppp_security_protocol** is set to **none**, the user will be prompted again for user name and passcode when trying to use the CLI **ppp** command. The user must enter the PIN and SecurID card code for the passcode.

   If you do not want to be prompted a second time, set **ppp_sec_auto** to **Y**.
4. **Set the RA 6300 parameters** password **and** vcli_password **and the port parameter** port_password **to the null string ("") if you want the ACE/Server system to authenticate all login attempts before allowing access to the RA 6300. Also, do not set a port password in the** acp_passwd **file when using SecurID.**
5. **Enter a host name or IP address for the** pref_secure1_host **and** pref_secure2_host **parameters for each RA 6300 using a SecurID card. The host addresses where each ACP process runs must be activated in the ACE/Server database as clients.**
6. **Set the RA 6300 parameter** security_broadcast **to** N **so that the RA 6300 does not inadvertently contact an ACP process that does user authentication via the** acp_passwd **file unless all the ACP server processes in your network are configured and installed to do user authentication by calling the ACE/Server.**

7. **Set the RA 6300 parameter** acp_key **to its assigned value and enter this value into the** acp_keys **file on the host . Then ACP and the RA 6300 exchange user names and passcodes encrypted with the key.**

8. **Activate valid RA 6300 users in the ACE/Server database with permissions (individual or group) to access the ACP servers. If two ACP servers are used, each user must be allowed access to both servers since either of them can authenticate a user by calling the ACE/Server host.**

Integrating SecurID into ACP

Integrating the ACE/Server software into ACP requires changes to the **erpcd** utility. The following instructions assume that the ACE/Server software is installed in a directory called **/usr/ace** and the RA 6300 software is installed in **/usr/annex**; if your code is installed in different directories, substitute the appropriate pathnames where applicable.

These instructions assume that the software is installed on a UNIX system and that the host tools have been compiled (as opposed to using the binaries from the RA 6300 distribution tape). Also, the target UNIX system requires a development environment (*C* compiler, libraries, etc.).

Make sure the host clock is set correctly.

1. **As a superuser, change into the** /usr/annex/src **directory:**

```
# cd /usr/annex/src
```

2. **Create a directory called** sdclient**:**

```
# mkdir sdclient
```

3. **Copy the required header files and libraries from the ACE/Server directories:**

   If you have ACE/Server Release 2.1.1 or 2.2:

   ```
   # cp /usr/ace/sdiclient.a sdclient
   # cp /usr/ace/prog/*.h sdclient
   ```

   This sequence requires that these files are installed on the slave/client system from the ACE/Server host.

   Make sure the ACE/Server UNIX Client is installed on the system that is running **erpcd**.

4. **Edit the** Makefile **file in the** /usr/annex/src/erpcd **directory:**

   ```
   # vi Makefile
   ```

5. **Kill the existing** erpcd **process (your process number will vary):**

   ```
   # ps -ax | grep erpcd
   25493 IW 0:00 ./erpcd
   25494 IW 0:00 ./erpcd

   25797 p1 S 0:00 grep erpcd
   # kill 25493
   ```

6. **Rebuild** erpcd **(see *Re-compiling erpcd* on page 15-556).**

7. **If you have linker errors try running the** ranlib **utility on the** sdiclient.a **library:**

   ```
   # ranlib sdclient/sdiclient.a
   ```

   Then rebuild **erpcd** (see *Re-compiling erpcd* on page 15-556).

8. **Make sure that ACP is enabled in the** eservices **file (the default is ACP disabled). The default file looks like this:**

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

Enable ACP by removing the pound sign (#) from its entry. The edited file looks like this:

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

9. **Run** erpcd **from the current directory or install the newly built** erpcd **in the** /usr/annex **directory by entering:**

```
# ./erpcd
```

or

```
# mv /usr/annex/erpcd /usr/annex/erpcd.old
```

```
# make install
# /usr/annex/erpcd
```

Now follow the procedures in the ACE/Server documentation for registering clients and users. The hosts where **erpcd** is running must be registered as clients, and all users with SecurID cards that will log into the RA 6300(s) must be allowed to access the host clients.

10. **On the RA 6300, enable security, configure the preferred security server, and enable CLI security on the ports to be protected by SecurID. If you have a secondary server, the new** erpcd **must be installed on that host and that host must be registered as a client in the ACE/Server database.**

A sample **admin** session looks like this:

```
admin:set annex enable_security Y security_broadcast N
admin:set annex pref_secure1_host calvin
admin:set port cli_security Y
admin:reset port
```

SecurID Backup Security

The RA 6300 uses the following procedures if the server running SecurID and **erpcd** is down:

- If the RA 6300 finds another server running **erpcd** but not SecurID, ACP will control RA 6300 security.
- If the RA 6300 cannot find another server running **erpcd**, the RA 6300 uses local security.

Using SafeWord AS Security

Enigma Logic’s SafeWord AS software verifies the identity of CLI users to permit access to protected systems. When you install SafeWord version 4.x on a central network server and link SafeWord to **erpcd**, SafeWord provides and authenticates fixed or dynamic passwords. SafeWord also supports the RADIUS server.

The difference in the application of SafeWord AS for this release is that the client/server approach now allows **erpcd** to communicate only with the SafeWord server through a client API. The server then interfaces with the database. Also, another difference is that clients are allowed to be on different hosts.

ACP hosts serve as clients to SafeWord AS.

You can use SafeWord software for:

- SLIP, PPP, IPX, and ARAP sessions only when you start a session from a CLI port. IPX users must connect from Fastlink II in terminal mode.

- The ARAP Remote Access client's CCL scripts in versions 1.0 and 2.0 as long as you do not use a SafeWord challenge as part of a dynamic password.

  ARAP does not use the **acp_regime** file.

- Macintosh or dial-back users only when the SafeWord user name matches the user name listed in the **acp_userinfo** file.

The RA 6300 supports SafeWord for user authentication only, therefore authorization is not supported. Therefore, when you dial in to the network through an RA 6300, or dial out from an RA 6300 (e.g., if you telnet to a port in slave mode), the RA 6300 does not display the SafeWord Failed Access Report. In addition, the RA 6300 does not run the user's SafeWord execute program at the end of the authentication process.

Installing SafeWord AS

To integrate SafeWord into ACP, you must make changes in the **erpcd** utility. You must install SafeWord:

- On a host running erpcd.
- On a UNIX system that has a development environment and compile the host tools (as opposed to using the binaries from the RA 6300 distribution tape).
- In a directory named **/safelog** and Remote Annex software in the **/usr/annex** directory. If you do not use these directory names, you must substitute pathnames.

Copy the following files during installation from the SafeWord AS installation directory into your **src/enigma** directory:

- NETWORKAPI/swecapi.h
- NETWORKAPI/swecapi.a
- LOCALAPI/custfail.h
- LOCALAPI/custpb.h

For a successful NETWORKAPI installation, you must also install the SafeWord AS client for UNIX machine (option 2 in the installation script) during the installation.

Makefile Switches

Define **Makefile** Switches

Define a new set of switches in the **Makefile** by uncommenting the following lines in **erpcd/Makefile**:

```
#ENIGMAFLAG=-DENIGMA_SAFEWORD -DNET_ENIGMA_ACP
#ENIGMAFILES=../enigma/swecapi.a
#ENIGMACFILES=acp_safeword.c
#ENIGMAOFILES=acp_safeword.o
```

"__assert" comes up undefined (the default). You must uncomment the following line as well:

```
#ENIGMAFLAG = -DENIGMA_SAFEWORD -DNET_ENIGMA_ACP
-DNEED_ENIGMA_ASSERT_PATCH
```

Configuration Management

Place a new file, called **safeword.cfg**, in the annex installation directory. This file is created as **sid.cfg** when you install the SafeWord AS client.

Move and Rename the **sid.cfg** File

To place the new file in the installation directory:

1. **Copy the** sid.cfg **file into the installation directory.**
2. **Rename the** sid.cfg **file to** safeword.cfg**.**

Create the **safeword.cfg** File

To create the **safeword.cfg** file, use the following example and replace *yourservername* with the name of your SafeWord server:

```
02 Authen. Server (host weight connects port):
     yourservername 0 0 7482

09 User ID Source (USER/SYSTEM): USER

10 Server’s System Name: STANDARD

15 Send Status Messages to User: NONE

16 Send Status Messages to Console: ERROR

17 Send Status Messages to log File: NONE

18 Status Message Log Filename: sid.log

23 Status Message Label: sid-7482
```

Integrating SafeWord into ACP

Before you use SafeWord, you need to integrate SafeWord into ACP:

1. **As a superuser, change to the** /usr/annex/src **directory:**

   ```
   # cd /usr/annex/src
   ```

2. **Create a directory called** enigma**:**

   ```
   # mkdir enigma
   ```

3. **Copy the** libidpb.a**,** custpb.h**, and** custfail.h **files into the enigma directory:**

   ```
   # cp /safelog/swecapi.a enigma/swecapi.h
   # cp /safelog/custpb.h enigma/custpb.h
   # cp /safelog/custfail.h enigma/custfail.h
   ```

   SafeWord's standard installation provides the **libidpb.a**, **custpb.h**, and **custfail.h** files in the /**safelog** directory.

4. **Edit the** make.config **file in the /annex/root/src directory:**

   ```
   # vi make.config
   ```

5. **Locate the following line, which is near the bottom of make.config:**

   ```
   CFG_STUBLINKING = -L. -lstubs
   ```

   Change the line to include the SafeWord library, as follows:

   ```
   CFG_STUBLINKING = ../enigma/libidpb.a -L. -lstubs
   ```

6. **If** erpcd **is running on the host, kill the existing** erpcd **process (your process number will vary):**

   ```
   # ps –ax | grep erpcd
   25493 ? IW 0:00 ./erpcd
   25797 p1 S 0:00 grep erpcd
   # kill -9 25493
   ```

7. **Rebuild** erpcd**:**

   ```
   # make erpcdIf you have linkage errors, try running
   the ranlib utility on the sdclient.a library:
   ```

   ```
   # ranlib enigma/lipidpb.a
   # make erpcd
   ```

8. **Install** erpcd **into the** usr/annex **directory:**

   ```
   # make install
   ```

9. **Restart** erpcd**:**

   ```
   # /usr/annex/erpcd
   ```

10. **On the RA 6300, use** admin **or** na **to** set pref_secure1_host **to the Internet address of the host running SafeWord and** erpcd**.**

    You can enter the backup host's address in the **pref_secure2_host** parameter.

SafeWord Passwords

SafeWord provides fixed and dynamic passwords to verify user access to protected systems.

While SafeWord's IDUTIL program allows administrators to create up to three levels of authentication for each user, RA 6300 access allows you to combine one dynamic and one fixed password: you cannot use two dynamic or two fixed passwords for a single authentication process.

If you configure a SafeWord startup file, it will not run when a user accesses an RA 6300.

Fixed Passwords

System administrators can generate a user's initial fixed password and can set the password's expiration date. When an existing password expires, RA 6300 users can choose a new fixed password:

1. **If the expiration message appears after you enter your username and password, press the** Escape **key and then press** Return.

   The *Old Fixed Password* message appears.

2. **Enter your old password and press** Return.

   The *New Fixed Password* message appears.

3. **Type your new password and press** Return.

   The *Repeat New Fixed Password* message appears.

4. **Type your new password again and press** Return.

   If you completed these steps correctly, the *Permission Granted* message appears. If you did not, SafeWord displays an error message.

Dynamic Passwords

SafeWord generates dynamic passwords using a hand-held password generator called a "token." The token generates new passwords each time a user wants to access protected systems. Network administrators can configure SafeWord's dynamic passwords in Synchronous, Semi-synchronous, and Asynchronous modes:

- In Synchronous mode authentication, the token generates a dynamic password that you enter at your terminal.
- In Semi-synchronous mode:
  - Enter the password from your previous session into the token, which then displays a new password.
  - Enter the dynamic password at your terminal.

- In Asynchronous mode:
  - The token displays a string, called a “challenge” before you enter a dynamic password.
  - Enter the challenge into the token, which generates a dynamic password.
  - Enter the dynamic password at your terminal.

You cannot use this (or any other authentication technique that uses a challenge) with PAP or IPX security, because neither PAP nor IPX allow challenges.

For detailed information about configuring and generating fixed and dynamic passwords, refer to Enigma Logic’s SafeWord documentation.

SafeWord Backup Security

The RA 6300 uses the following procedures if the server running SafeWord and **erpcd** is down:

- If the RA 6300 finds another server running **erpcd** but not SafeWord, ACP will control RA 6300 security.
- If the RA 6300 cannot find another server running **erpcd**, the RA 6300 uses local security.

Configuring Security for the Remote Annex FTP Daemon

When a new FTP session is initiated, the FTP daemon registers the source host with the **who** database. A subsequent **who** displays:

```
annex: who

 Port    What    User    Location    When    Idle  Address
 1       PSVR    ---     jdcm        4:06am  3:13  192.9.200.60
                         console
 v1      CLI     hobbes  ---         4:07am        192.9.200.60
 v2      FTPD    ---     ---         ---     :01   bryce
 annex:
```

In the above sample command display, since the user has not yet logged into the **ftp** session, no user name appears in the *User* field. If the **enable_security** parameter is set to **Y** but a preferred security server is not configured, or if **enable_security** is set to **N**, the user is prompted for a user name and a password. The Remote Annex will accept any user name, but grants FTP access only after checking the password against its administrative password. If the Remote Annex grants access, the user's name appears in the **who** command display.

If the **enable_security** parameter is set to **Y** and a preferred security server is configured, the Remote Annex calls the **ppp_security** function in the **acp_policy.c** file with the user's name and password as entered and the service set to SERVICE_FTP. If ACP grants access, the FTP daemon will ask for an "account." The Remote Annex compares the text entered at this prompt against its administrative password for an added level of security.

If the **enable_security** parameter is set to **Y** and the preferred security server is not reachable, the Remote Annex denies access to the FTP daemon.

When the validation process is complete, the Remote Annex logs FTP access in the ACP logfile (see *Host-based Security Logging* on page B-26) and updates the **who** command display to look something like this:

```
annex: who

 Port   What   User    Location    When   Idle  Address
 1      PSVR   ---     jdcm        4:06am 3:37  192.9.200.60
                       console
 v1     CLI    hobbes  ---         4:07am       192.9.200.60
 v2     FTPD   hobbes  ---         2:43pm       bryce
 annex:
```

The Remote Annex FTP daemon is compatible with all versions of UNIX **ftp.**

You can completely disable the Remote Annex FTP daemon by setting **ftpd** in the **disabled_modules** parameter.

Configuring the IP Basic Security Option (IPSO)

The Department of Defense Basic Security Option for IP identifies the U.S. classification level at which an IP datagram is to be protected and the authorities whose protection rules apply to each datagram, as defined in RFC 1108.

The Remote Annex partially implements this security option by adding the IPSO classification level to packets generated by **telnet** or **rlogin** running on a Remote Annex dedicated, adaptive, or CLI port. (The CLI port can be an **auto_detect** or **auto_adapt** port that the user has put into **cli mode** by pressing **Return** when first connected to the port.) The Remote Annex does not add the option to locally generated system packets, such as ICMP messages and RIP updates. Nor does the Remote Annex check incoming packets for the presence of the IP Security Option.

To set the IPSO for packets generated on a port:

1. **Use the** na **utility, the superuser CLI** admin **command, or SNMP to set the Remote Annex parameter** enable_security **to** Y **(the default is** N**).**
2. **Use** na, admin**, or SNMP to set the serial line port parameter** ipso_class **to one of the following values:** topsecret**,** secret**,** confidential**,** unclassified**, or** none**. If you specify** none **(the default), the Remote Annex does not add the option to packets.**

The **ipso_class** parameter is also an object in the private-enterprise MIB and can be set via SNMP (for more details, see *Simple Network Management Protocol (SNMP)* on page B-41).

The following sample **su** session causes a basic security option of secret to be included in all packets generated by ports 1 and 2.

```
annex: su
Password:
annex# admin
Annex administration Remote Annex Rx.x, 72 ports
admin: set port=1,2 ipso_class secret
admin: set port mode cli
admin:
```

When a router that fully implements IPSO receives a packet with an unacceptable classification level, it sends an ICMP security discard message to the packet's originator. If the Remote Annex receives a discard message, it passes it to the application running on the port that generated the IPSO packet.

Logging Security Events

Host-based security can 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 in an ACP log file.

The ACP log file can be the default **acp_logfile** located in the **/usr/annex** directory or a Remote Annex-specific log file. A Remote Annex-specific log file is created by uncommenting the following statement in the **acp_policy.h** file:

```
#define SEPARATE_LOGS
```

Once this statement is uncommented, a Remote Annex-specific log file is created with the name **acp_logfile.*Annex_IPaddress*** in the **/usr/annex** directory.

Each logged message in the ACP log file contains the following fields:

- *IP address of the Remote Annex*
- *Sequence number*
- *Port number*
- *Date*
- *Time*
- *Module*
- *Event*
- *Packets in*
- *Packets out*
- *Bytes in*
- *Bytes out*
- *Protocol-dependent information*
- *Username*

All fields are separated by colons and are encoded for use by UNIX utilities that sort, merge, select, or filter streams. *Host-based Security Logging* on page B-26 provides a sample log file.

The parser of the **acp_userinfo** file generates log messages if an error is detected when processing a user's profile.

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**.

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 re-compile **erpcd**, kill the current version, and start the new version (see *Modifying the Code* on page 15-556).

Disabling User Name and Password Validation

When security is enabled, users must provide a user name and password. You can disable this policy by modifying the */annex_root/***src/erpcd/acp_policy.h** file.

To disable the user name requirement, 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 the message does not include a user name.

To disable the port password requirement, make sure the following line is commented out (i.e., enclosed in asterisks), as follows:

```
/* #define PORT_PASSWORD 1 */
```

Linking NIS Password File Verification to ACP

You can enable several options in the **acp_policy.h** file by removing the slash (/) and asterisk (*) at the beginning and the end of the definition line.

To use the NIS password file for verification through ACP, change (uncomment) the following lines:

```
/* #define NATIVEPASSWD 1 */ to #define NATIVEPASSWD 1
/* #define NATIVESHADOW 1 */ to #define NATIVESHADOW 1
```

You can change several other options in the same way:

```
/*
     * Uncomment this line to select the use of the\
     * standard syslog(3) facility in addition to or in\
     * place of the logfile -- the value of "USE_SYSLOG"\
     * is used to identify the daemon.(Comment the
     * second line out to disable the normal acp log file.)
*/
/* #define USE_SYSLOG "annex" */
#define USE_LOGFILE 1

/*
     * Uncomment this line to use decoded Annex peer names,\
     * rather than numeric IP addresses, in the log file
     * and in syslogging.
*/
/* #define USE_ANAME 1 */
```

Modifying Message Formats in the ACP Log File

The USE_SECONDS option in the **acp_policy.h** file enables messages in the ACP log file to use a *seconds-since-1970* (ten decimal digits) format. This format is most useful for automatic ACP log file parsing programs since these programs frequently need to do comparisons and arithmetic on dates. This option is disabled by default.

You can enable USE_SECONDS by changing (uncommenting) the following line:

```
/* #define USE_SECONDS 1*/ to #define USE_SECONDS 1
```

The standard message format in the ACP log file is:

```
<annex_name>:<logid>:#<port>:<yymmdd>:<hhmmss>:<service>:\
<event>:<pkts in>:<pkts out>:<bytes in>:<bytes out>:<msg>
```

When USE_SECONDS is enabled, the message format in the ACP log file is:

```
<annex_name>:<logid>:#<port>:<seconds_since_1970>:\
<service>:<event>:<pkts in>:<pkts out>:<bytes in>:\
<bytes out>:<msg>
```

Changing the Expected File Names Used by ACP

The supplied policy uses 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 the **acp_passwd** file, change the following lines in the **acp_policy.h** file:

```
#define ACP_PASSWD (str) \
   sprintf(str,"%s/acp_passwd",install_dir)

#define ACP_PTMP (str) \
   sprintf(str,"%s/acp_ptmp",install_dir)
```

To change only the filename:

```
#define ACP_PASSWD (str) \
   sprintf(str,"%s/new_filename",install_dir)

#define ACP_PTMP (str) \
   sprintf(str,"%s/new_tempfile",install_dir)
```

To change the full pathname:

```
#define ACP_PASSWD (str) \
   sprintf(str,"new_path/new_filename")

#define ACP_PTMP (str) \
   sprintf(str,"new_path/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.

The **install_dir** is defined in the file */annex_root/***src/make.config** with the leading quote supplied by the makefile. Since the trailing quote is required by the two strings, double quote the names for the new password and temporary files.

You can change the names of several other files in the **acp_policy.h** file in the same way:

```
#ifdef NATIVESHADOW

#define ACP_SHADOW(str)\
     strcpy(str,"/etc/shadow")
#define ACP_STMP(str)\
     strcpy(str,"/etc/shadow.tmp")
#define ACP_LOCKFILE(str)\
     strcpy(str,"/etc/.pwd.lock")
#define ACP_GROUP(str)\
     strcpy(str,"/etc/group")
#else
```

(continued on next page)

```
#define ACP_SHADOW(str)\
     sprintf(str,"%s/acp_shadow",install_dir)

#define ACP_STMP(str)\
     sprintf(str,"%s/acp_stmp",install_dir)

#define ACP_LOCKFILE(str)\
     sprintf(str,"%s/.pwd.lock",install_dir)

#define ACP_GROUP(str)\
     sprintf(str,"%s/acp_group",install_dir)

#endif

/*   define pathname of accounting file*/

#define ACP_LOGFILE(str) \

     sprintf(str,"%s/acp_logfile",install_dir)

/*   define pathname for restrictions file*/

#define ACP_RESTRICT(str) \

     sprintf(str,"%s/acp_restrict",install_dir)

/*   define pathanme for annex acp_keys file */

#define ACP_KEYS(str) \

     sprintf(str,"%s/acp_keys",install_dir)

/*   define pathanme for annex dialup addresses file */

#define ACP_DIALUP(str) \

     sprintf(str,"%s/acp_dialup",install_dir)

/*   define pathname for user profile file */

#define ACP_USERINFO(str) \

     sprintf(str,"%s/acp_userinfo",install_dir)

#define ACP_ESERVICES(str) \

     sprintf(str,"%s/eservices",install_dir)
```

In the same way, you can also change the expected prompts for default applications:

```
#ifndef SECURID_CARD

#define ACP_USERPROMPT "Annex username: "
#define ACP_PASSPROMPT "Annex password: "
#define ACP_PERMGRANTD "\nPermission granted\n"
#define ACP_PERMDENIED "\007\nPermission denied\n"
#define ACP_INCORRECT "\nUsername/Password Incorrect\n"

#else

#define ACP_USERPROMPT "Username: "
#define ACP_PASSPROMPT "Enter PASSCODE: "
#define ACP_PERMGRANTD "\nPASSCODE accepted\n"
#define ACP_PERMDENIED "\007\nAccess Denied\n"
#define ACP_INCORRECT "\nUsername/PASSCODE Incorrect\n"

#endif

#define ACP_TIMEDOUT "\007\nLogin Timed Out\n"
#define ACP_WARNING "\007\nYour password will expire \
     in %ld days unless changed.\n"
#define ACP_WARNINGM "\007\nYour password expires after\
     tomorrow unless changed.\n"
#define ACP_WARNINGT "\007\nYour password expires after \
     today unless changed.\n"

#define ACP_AWARNING "\007\nYour account will expire in\
     %ld days.\n"
#define ACP_AWARNINGM "\007\nYour account expires after\
     tomorrow.\n"
#define ACP_AWARNINGT "\007\nYour account expires after\
     today.\n"
#define ACP_EXPIRED "Your password has expired.\n"
#define ACP_NEWPASS "Enter a new password:  "
#define ACP_NEWPASS2 "Re-enter new password:  "

#define ACP_PASSMATCH "Entered passwords do not match.\
     Try again.\n"
#define ACP_ACCESSCODEPROMPT "Access Code: "
#define ACP_PHONEPROMPT "Telephone Number: "
#define ACP_DIALBACKGRANTD "\nRequest accepted,dialback in\
     progress\n"
#define ACP_CLINODIALBACK "\nPermission granted, no\
     dialback\n"
```

(continued on next page)

```
/*   define messages used by Securid Card application*/

#ifdef SECURID_CARD
#define ACP_NEXTCODEPROMPT "Enter next card code: "
#define ACP_PINCHAR "characters"
#define ACP_PINDIGIT "digits"
#define ACP_PINSIZE "%d"
#define ACP_PINSZRANGE "%d to %d"
#define ACP_NEWPINPROMPT "Enter your new PIN containing %\
     %s,\n"
#define ACP_OR "\t\tor\n"
#define ACP_NEWPIN_2 "Press Return to generate new PIN and\
     display it\n"
#define ACP_NEWPIN_3 "<Ctrl d> to leave your card in New-PIN\
     mode.\n"
#define ACP_SYSGENPIN "\t\t%s\n"
#define ACP_PINREENTRY "Please re-enter PIN: "
#endif#ifdef PORT_PASSWORD
/* only if PORT_PASSWORD is set and a port password exists
in acp_passwd */
#define ACP_PORTPROMPT "Port password: "
#endif

/* miscellaneous defines for default application */

#define INPUT_TIMEOUT 30
#define INPUT_POLL_TIMEOUT 3
#define RETRIES_MAX 3
```

Locking the ACP Log File

To prevent two or more hosts processes from logging a record simultaneously, the Remote Annex **erpcd** code uses the host system call **lockf** to lock the ACP log file. This lock prevents other processes from writing the file until the file update is complete.

There are two ways to use the system **lockf** call. You can select either mechanism via a switch in the **acp_policy.h** file. The following explanation of the switch resides in this file. The default method, T_LOCK, is reliable but not very efficient; F_LOCK is more efficient but does not work on all hosts (some host manufacturers have issued patches that resolve this issue).

```
/*
* Uncomment this line to select the F_LOCK method to lock the
* ACP log file for updating.
*
* A file must be locked for update in order to block other
* processes from writing to it simultaneously.
*
* F_LOCK - Passing the F_LOCK as the cmd value when making
* system lockf call is the most efficient and preferred manner
* to lock a file for exclusive write access. In this scenario
* a process is put to sleep until the resource is available.
* Once available the process is preempted owning the resource.
*
* T_LOCK - When the T_LOCK cmd argument is passed, the process
* must repeatedly send the lockf call the until the resource
* is available.Once available the system call returns a
* success and the resource is acquired.
*
* The F_LOCK cmd has been determined to be faulty on many
* hosts. Failures can not be narrowed down to any particular
* hardware manufacturer or UNIX system. There are to many OS
* revs and variables to sense the correct lockf method to
* use at installation time. The default, T_LOCK was chosen
* simply because it has been proven reliable. SEE
* 'log_message()'
*/
/* #define USE_F_LOCK 1 */
```

Masking CLI Commands

When the security subsystem is enabled, you can mask (disable) user access to specific CLI commands by modifying the CLI_MASK line in the **acp_policy.h** file.

To disable **rlogin** and **telnet** for all users that enter the system through ACP security, modify the definition line to read:

```
#define CLI_MASK (unsigned long) (MASK_RLOGIN | MASK_TELNET)
```

To disable the CLI **who** and **su** commands for all users that enter the system through ACP security, modify the definition line to read:

```
#define CLI_MASK (unsigned long) (MASK_WHO | MASK_SU)
```

You can extend this to any set of commands by adding masks to that line separated by the vertical bar (|).

If the user enters the masked command, the CLI displays an error message. The superuser CLI commands cannot be masked individually. They can all be disabled by masking the **su** command.

Superuser CLI mode overrides ACP command masking.

You can disable several other CLI commands in the same way:

```
/*   define bit to disable each maskable CLI command*/

#define MASK_BG       0x00000001
#define MASK_CALL     0x00000002
#define MASK_FG       0x00000004
#define MASK_HANGUP   0x00000008
#define MASK_HELP     0x00000010
#define MASK_HOSTS    0x00000020
#define MASK_JOBS     0x00000040
#define MASK_KILL     0x00000080
#define MASK_NETSTAT  0x00000100
#define MASK_RLOGIN   0x00000200
#define MASK_STATS    0x00000400
#define MASK_STTY     0x00000800
#define MASK_TELNET   0x00001000
#define MASK_WHO      0x00002000
#define MASK_LOCK     0x00004000
#define MASK_SU       0x00008000
#define MASK_SLIP     0x00010000
#define MASK_CONNECT  0x00020000
#define MASK_SERVICES 0x00040000
#define MASK_PPP      0x00080000
#define MASK_ARAP     0x00100000
#define MASK_NONE     0x80000000
```

After changing the code, **cd** to the **/src** directory and recompile **erpcd**.

For more specific command disabling, e.g., by user name, you must edit the distribution policy file */annex_root/***src/erpcd/acp_policy.c**.

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 starts a new version of itself each time a security request is received from a Remote 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 **acp_policy.c** is documented in the form of *C* programming language comments. The file **policy.doc** provides a complete description of the available library functions.

Re-compiling erpcd

You must re-compile **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 Remote Annex's source code was copied. To re-compile:

1. **cd to** /annex_root/src**.**
2. **To re-compile only** erpcd**, enter the command:**

   ```
   # make erpcd
   ```

3. **To re-compile both** erpcd **and** ch_passwd**, enter the command:**

   ```
   # make 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.**

Using the ch_passwd Utility

The **ch_passwd** utility enables users to change their passwords when accessing a Remote Annex through the Access Control Protocol (ACP) security system. This utility affects only passwords in the **acp_passwd** or **acp_shadow** file. Table A-73 describes the supported argument for **ch_passwd**.

To change a Remote Annex user password, the *username* in the **acp_passwd** file must match the *username* in the **/etc/passwd** (or /**etc/shadow**) file on the ACP host.

If ACP is configured to record password histories, it saves the passwords set via the **ch_passwd** command. ACP keeps these passwords in the **acp_dbm** database on the security host, keyed by user name. The value of the STORED_PASS variable in **acp_policy.h** determines the number of passwords saved. This variable is initialized to 6 for **passwd**/**shadow** files and 0 for **passwd** files alone. A value of 0 disables password history. For more information, see *Enabling and Configuring Password Histories* on page 15-488.

The **ch_passwd** utility first prompts for the old password, and then for the new one. The syntax is:

ch_passwd

A superuser can change the password for any user. The syntax is:

ch_passwd [*username*] [**–s** *directory*]

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 Remote Annex software distribution and are located in the */annex_root*/**src/erpcd** directory. For instructions on recompiling both, see *Configuring Hosts and Servers* on page A-343.

Table A-73. Supported Argument for ch_passwd

Argument	Description
–s *directory*	Specifies the directory for the security files (**acp_passwd** and, if configured, **acp_shadow**); defaults to the defined install-annex directory (usually **/etc/annex/**).