CHAPTER 21

Configuring Protocol Translation

This chapter describes how to configure protocol translation connections on the communication server. It assumes you understand how to use the configuration software. It provides procedures for specifying system-wide facilities, as well as application examples. Before continuing with this chapter, be sure that you are familiar with the information provided in the X.25, Telnet, LAT, TN3270, and XRemote configuration chapters in this publication.

For a complete explanation of the **translate** command, refer to the *Access and Communication Servers Command Reference* publication. For more information about making connections and establishing translation sessions, see the *Cisco Access Connection Guide*.

In the context of this chapter, a communication server set up to run protocol translation software is referred to as a protocol translator.

Note Telnet is a remote terminal protocol that is part of the Transmission Control Protocol/Internet Protocol (TCP/IP) suite. The descriptions and examples in the following sections use the term TCP as a reference to Telnet functionality.

Support for Terminal Connections and Protocol Translation

A communication server set up for protocol translation uses the International Telecommunication Union Telecommunication Standardization Sector (ITU-T) Recommendation X.25 for transferring raw data over X.25 networks. The X.25 software supports both commercial and DDN versions. Protocol translators also support X.25 as a transport mechanism for IP packets, and X.3- and X.29-based PAD connections. This allows the protocol translators to connect to an X.25 public data network (PDN). This X.25 connection allows transport of TCP/IP packets across the X.25 packet-switching network in the same way as would occur on a protocol translator.

Note The ITU-T carries out the functions of the former Consultative Committee for International Telegraph and Telephone (CCITT).

Communication servers without protocol translation do not support an X.25 PAD function, so they cannot communicate with hosts directly connected to the X.25 PDN. Communication servers encapsulate TCP/IP packets in X.25 packets for transfer over a packet-switching network. These packets can be received by communication servers. Only servers with protocol translation include PAD capabilities, but other servers can communicate with protocol translators using the TCP/IP or LAT protocols. Server products with protocol translation can translate TCP/IP and LAT into X.25, and then communicate with X.25 hosts. Server products with protocol translation support all PAD standards (X.28, X.29, and X.3). The connection to the packet-switching network is through a synchronous line.

Connections to a PAD are made using EXEC commands. You can configure PAD parameter profiles that can be used to set PAD parameters with other commands, and you can configure access lists to control X.25 network access. Both these features use the message fields defined in Recommendation X.29, which describes procedures for exchanging data between PADs or between a PAD and a DTE.

Change the Number of Supported Translation Sessions

IOS Release 10.3 software supports a maximum of 100 sessions on a communication server with protocol translation enabled, and 32 sessions if routing is also enabled.

Because each protocol translation session uses a virtual terminal line, you need to increase the number of those lines to increase the number of protocol sessions. That is, if your communication server has ten virtual terminal lines, you can have up to ten protocol translation sessions. The default number of virtual terminal lines is 5 (lines 0 through 4). To increase the number of lines, and thus the maximum number of protocol translation sessions, perform the following tasks, as appropriate, in line configuration mode:

Task	Commands
Increase the number of protocol translation sessions.	**line vty** *line-number*
Decrease the number of protocol translation sessions.	**no line vty** *line-number*

Increasing the number of protocol translation sessions while routing is enabled can impact memory. The amount of memory available depends on the platform type, the amount of DRAM available, the activity of each session, and the speed of the link. If you are using the maximum number of sessions and have problems with memory, you might need to decrease the number of protocol translation sessions.

Cisco's Implementation of Protocol Translation

The protocol translation software attempts to provide transparent protocol translation between systems running disparate protocols. It enables terminal users on one network to access hosts on another network, despite differences in the native protocol stacks associated with the originating device and the target host.

A communication server supports virtual terminal connections in both directions between the protocols in the following list. You can configure the communication server to translate automatically between them. This is called the one-step translation method.

- X.25 and Local Area Transport (LAT)
- X.25 and Telnet sessions using the Transmission Control Protocol (TCP)
- LAT and TCP/Telnet

You can also use the one-step protocol translation facility to tunnel SLIP and PPP inside of X.25, LAT, or TCP/Telnet (on outgoing connections only).

The software supports limited connections in both directions between the following protocols. Connecting between these protocols requires that you first connect to a communication server running protocol translation, then to the host to which you want to connect. This is called the two-step translation method.

- XRemote to X.25 PAD environments (XRemote must use the two-step method)
- TN3270 to LAT, X.25, and TCP/Telnet (TN3270 must use the two-step method)

The following sections describe the process of tunneling SLIP and PPP using protocol translation, as well as the two-step and the one-step translation methods. See the end of this chapter for protocol translation application and session examples.

Understand Tunneling of SLIP and PPP

While other protocols, such as LAT, X.25, and TCP, are actually translated when you use one-step or two-step protocol translation, SLIP and PPP are not translated to the destination protocol. Instead, they are tunnelled inside the LAT, X.25, or TCP packets specific to the device on the remote network. Nonetheless, you use the protocol translation facility to tunnel SLIP or PPP to a destination running a different virtual terminal protocol (such as X.25).

If you want to use one-step protocol translation to tunnel SLIP or PPP, you do not need to enter any special commands. Simply use the **translate** command with the SLIP or PPP keywords for one-step connections. Refer to the section "Understand the One-Step Method" in this chapter for more information about one-step protocol translation.

To tunnel SLIP or PPP using the two-step protocol translation method, use the **vty-async** command, which enables you to run SLIP and PPP on virtual terminal lines. Normally, SLIP and PPP function only on physical asynchronous interfaces. Protocol translation, however, occurs on virtual terminal lines, which can be created and eliminated dynamically using the **line vty** command. The **vty-async** command enables asynchronous protocol functionality on virtual terminal lines, which permits tunnelling of SLIP and PPP using the protocol translation facility.

For more information about enabling asynchronous protocol functionality on virtual asynchronous interfaces, refer to the section "Enable SLIP and PPP on Virtual Asynchronous Interfaces" in the chapter "Configuring SLIP and PPP" in this publication.

If you are tunneling PPP or SLIP across X.25, you must also set up your X.3 profile correctly using the **x25 profile** command, as described in the section "Configure One-Step Tunneling of SLIP or PPP" later in this chapter. For more information about using the **x25 profile** command, refer to the chapter "Protocol Translation Configuration Commands" in the *Access and Communication Servers Command Reference*.

Note You must ensure that if the PAD is not configured to use the X.3 parameters by default that you specify using the **pad** [/**profile** *name*] command, it must be set to accept parameter changes from the X25 host.

Understand the Two-Step Method

In general, you use the two-step process when you want to use protocol translation for one-time connections or when you want to use a communication server running protocol translation as a general-purpose gateway between two types of networks (for example, X.25 PDN and TCP/IP). You must first configure the communication server for the transmission protocols that you will be using.

Note You must use the two-step method for translations of TN3270 and XRemote.

With the two-step connection process, you can modify the parameters of either network connection, even while a session is in process. This process is similar to connecting a group of terminal lines from a PAD to a group of terminal lines from a TCP server. The difference is that you do not encounter the wiring complexity, unreliability, management problems, and performance bottlenecks that occur when two devices are connected via asynchronous serial lines.

Also, the two-step process allows another level of security over the one-step translation method when TACACS and password protection are enabled. These security features are described in the "Managing the System" chapter in this publication.

Understand the One-Step Method

In general, you use the one-step method when network users repeatedly log on to the same remote network hosts through a communication server. This connection is more efficient and enables the communication server to have more knowledge of the protocols in use because the communication server acts as a network connection rather than as a terminal.

The one-step method provides transparent protocol conversion. When connecting to the remote network host, the user enters the connection command to the remote network host, but does not need to specify protocol translation. The network administrator has already created a configuration that defines a connection and the protocols to be translated. The user performs one step to connect with the host.

When you make a one-step connection to the communication server, the communication server determines which host the connection is for and which protocol that host is using. It then establishes a new network connection using the protocol required by that host.

A disadvantage of the one-step method is that the initiating computer or user does not know that two networking protocols are being used. This means that parameters of the foreign network protocols cannot be changed after connections are established. The exception to this limitation is any set of parameters common to both networking protocols. Any parameter common to both can be changed from the first host to the final destination.

To configure the one-step method of protocol translation, set up the following protocols and connection options in the configuration file:

- The incoming connection—The configuration includes the protocol to be used—LAT, X.25, or TCP/IP (Telnet)—the address, and any options such as reverse charging or binary mode that are supported for the incoming connection.
- The outgoing connection—The outgoing connection is defined in the same way as the incoming connection, except that SLIP and PPP are now also supported.
- The connection features global options

  You can specify additional features for the connection that allows, for example, incoming call addresses to match access-list conditions or that limit the number of users that can make the connection.

Configure Protocol Translation

Specifically, this section describes how to perform the following tasks:

- Configure Two-Step Protocol Translation
- Configure One-Step Protocol Translation

Configure Two-Step Protocol Translation

To translate using the two-step method, perform the following tasks beginning in global configuration mode. The first step is required only if you are tunnelling SLIP or PPP using the two-step protocol translation facility:

Task		Command
Step 1	Establish an incoming connection to the communication server running protocol translation.	{**connect** | **lat** | **pad** | **telnet** | **tunnel**}[1]
Step 2	Establish the outgoing connection from the communication server running protocol translation to another network host.	{**connect** | **lat** | **pad** | **telnet** | **tunnel**}[1]

1. Each of these commands is described in the *Cisco Access Connection Guide*.

Protocol translation software supports the two-step method in both directions for protocols other than SLIP and PPP (for example, from Telnet to PAD, and vice versa). SLIP and PPP are only supported on outgoing connections.

Configure One-Step Protocol Translation

To create one-step protocol translation connection specifications, perform the following task in global configuration mode:

Task	Command
Create the connection specifications for one-step protocol translation.	**translate** *protocol incoming-address* [*in-options*] *protocol outgoing-address* [*out-options*] [*global-options*]

Define X.25 Host Names

This section describes how to define symbolic host names. To define a symbolic host name, perform the following task in global configuration mode:

Task	Command
Define a symbolic host name.	**x25 host** *name x.121-address* [**cud** *call-user-data*]

Protocol Translation Application Examples

This section provides protocol translation examples for the following applications:

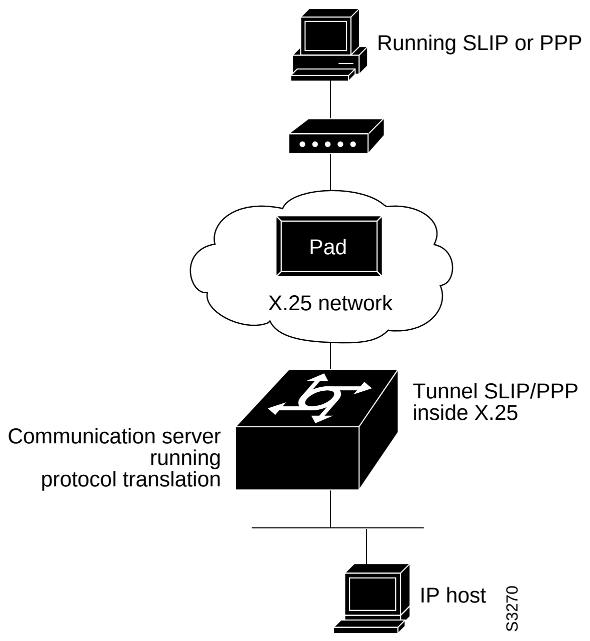
- Tunnelling PPP across X.25 Example
- Tunnelling SLIP in TCP Example
- Local LAT-to-TCP Translation Example
- LAT-to-TCP Translation over a WAN Example
- LAT-to-LAT Translation over a WAN Example

Note In the application illustrations that follow throughout the remainder of this chapter, source and destination device icons used to illustrate the flow of translated information are shown with black type in outlined shapes. Other elements in the environment are shown with reverse type on solid black shapes.

Tunnelling PPP across X.25 Example

A communication server can tunnel PPP traffic across an X.25 WAN to allow communication among resources in these protocol environments. In Figure 21-1, the PC establishes a PPP session with the communication server through an X.25 network using CHAP authentication.

Figure 21-1 Tunnelling SLIP or PPP in X.25


Running SLIP or PPP
Pad
X.25 network
Tunnel SLIP/PPP inside X.25
Communication server running protocol translation
IP host
S3270

The following configuration tunnels PPP over X.25 from the PPP client to the virtual asynchronous interface with IP address 10.0.0.4. Routing and CHAP authentication are enabled for the PPP session. The X.121 address of the X.25 host is 31370054065. An X.29 profile script names *x25-ppp* is created using the following X.3 PAD parameters:

- 1:0, 2:0, 3:2, 4:1, 5:0, 6:0, 7:21, 8:0, 9:0, 10:0, 11:14, 12:0, 13:0, 14:0, 15:0, 16:127, 17:24, 18:18, 19:0, 20:0, 21:0, 22:0

For more information about X.3 PAD parameters, refer to the appendix “X.3 PAD Parameters” in the *Remote Node and Terminal Services Command Reference*. If you were performing a two-step connection, you would specify these X.3 PAD parameters using the **pad** [/**profile** *name*] command, which is described in the chapter “Remote Node and Terminal Connections using Protocol Translation” in the *Cisco Access Connection Guide*.

With the router connected to IP host, the PC running PPP can now communicate with the IP host.

```
2509# config term
```

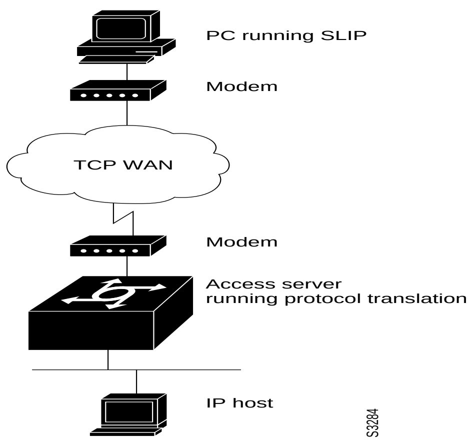
```
2509(config)# X29 profile x25-ppp 1:0  2:0  3:2  4:1  5:0  6:0  7:21  8:0  9:0
              10:0  11:14  12:0  13:0  14:0  15:0  16:127  17:24 18:18
2509(config)# translate x25 31370054065 profile x25-ppp ppp 10.0.0.4 routing
              authentication chap
```

This is only a partial example. The commands in this example would only be a part of the complete configuration file for an individual device.

Tunnelling SLIP in TCP Example

A communication server running protocol translation software can translate between TCP and SLIP traffic to allow communication among resources in these protocol environments. In Figure 21-2, the PC running SLIP is connecting to a TCP/IP network and making a connection with the device IP host. This example enables routing and turns on header compression.

Figure 21-2 Tunneling SLIP in TCP Example



The following configuration tunnels SLIP inside of TCP packets from the SLIP client with IP address 2.0.0.5 to the communication server. The communication server then establishes a protocol translation session to IP host. Routing and header compression are enabled for the SLIP session.

```
translate tcp 1.0.0.1 slip 2.0.0.5 routing header-compression passive
```

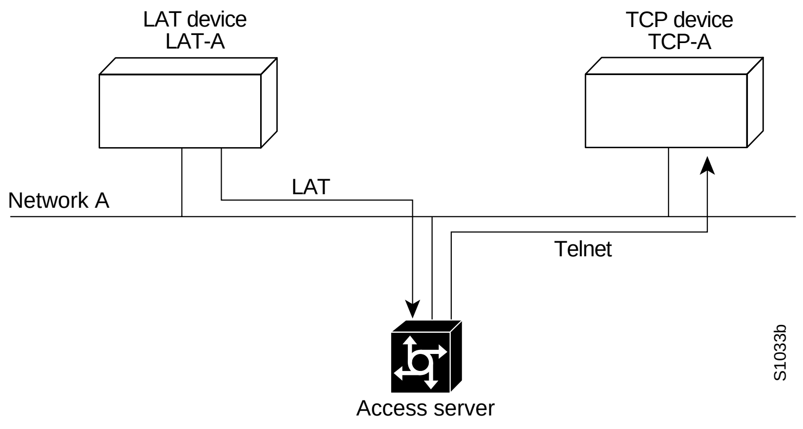
The device IP host on a different network attached to the communication server can be accessed by the SLIP client because routing has been enabled on the interface in the communication server where the SLIP session is established.

This is only a partial example. The commands in this example would be only part of the complete configuration file for an individual device.

Local LAT-to-TCP Translation Example

Figure 21-3 shows a simple LAT-to-TCP translation across an Ethernet network. The configuration file for the communication server follows the figure. The name *TCPA* is the logical name given to the device *TCP-A*.

Figure 21-3 Local LAT-to-TCP Translation



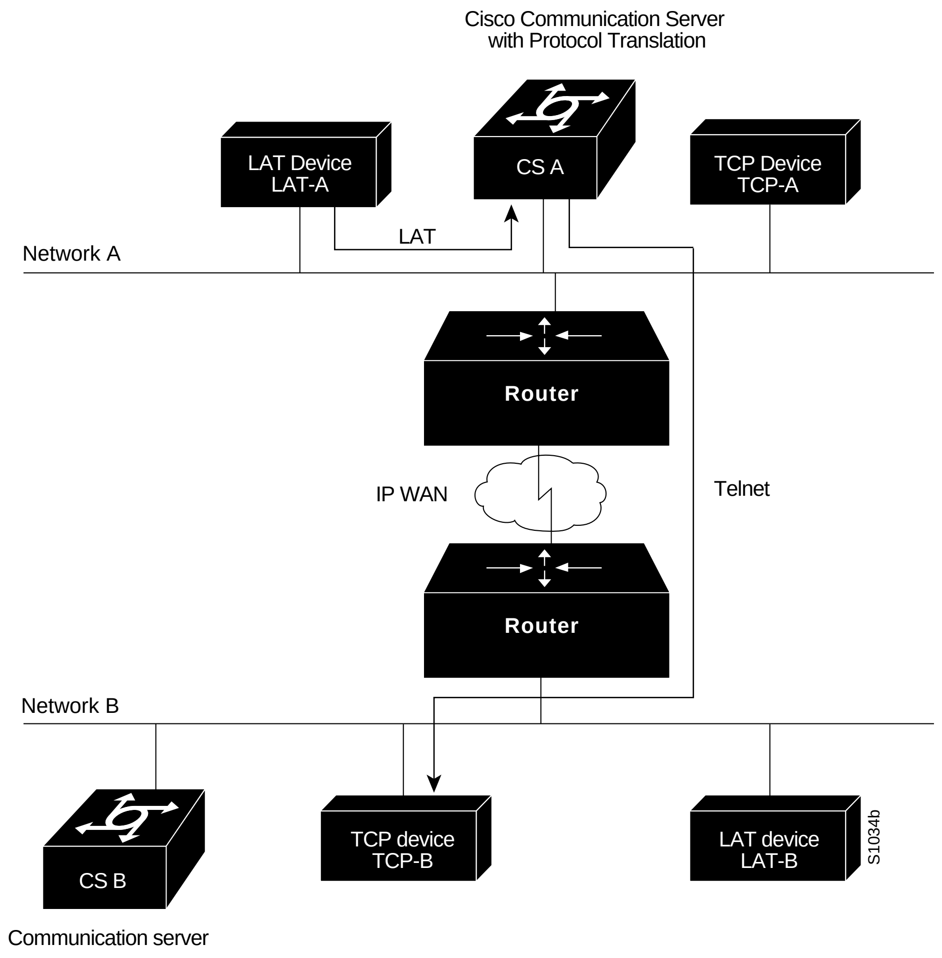
Configuration for Communication Server

```
interface ethernet 0
ip address 1.0.0.2 255.255.0.0
!
! enable LAT on this interface
lat enabled
!
translate lat TCPA tcp TCP-A
```

LAT-to-TCP Translation over a WAN Example

Figure 21-4 shows a configuration that allows translation of LAT to TCP and transmission across an IP-based WAN. The configuration file for CS A follows the figure. The logical LAT service name *distant-TCP* is the name given to device *TCP-B*.

Figure 21-4 LAT-to-TCP Translation over a WAN



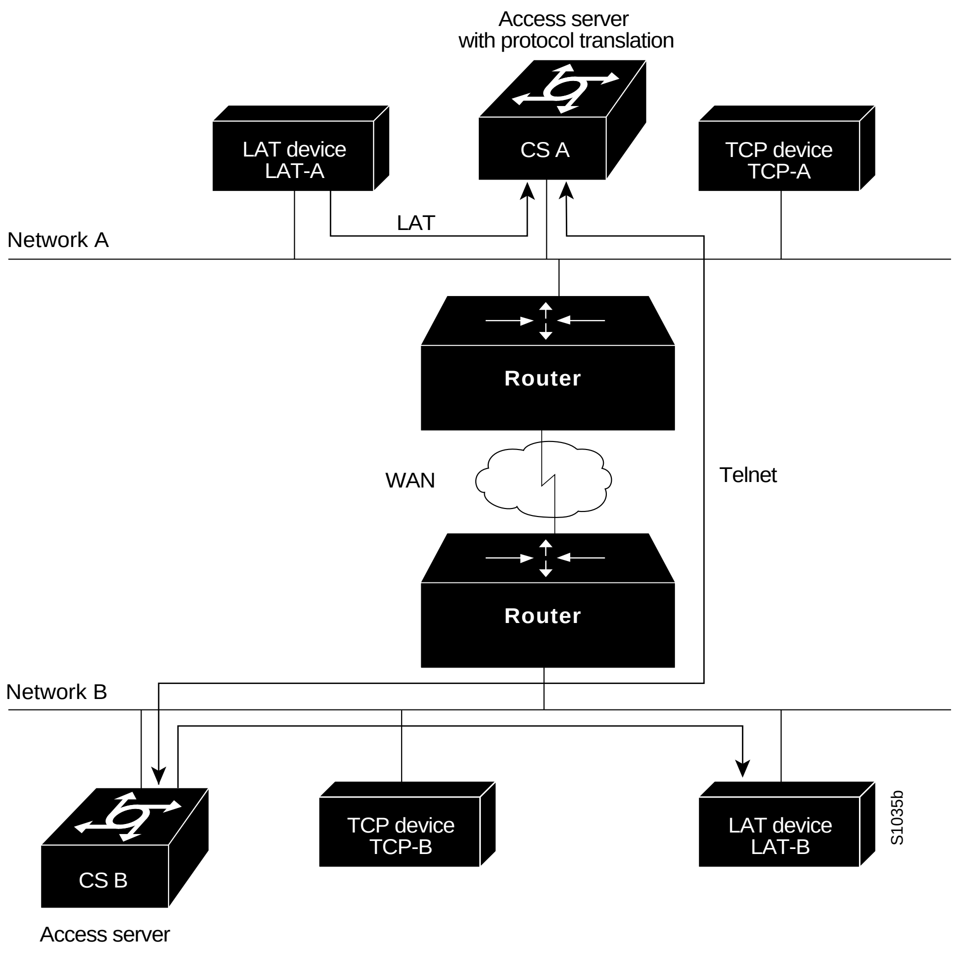
Configuration for CS A

```
interface ethernet 0
ip address 1.0.0.2 255.255.0.0
!
! enable LAT on this interface
lat enabled
!
translate lat distant-TCP tcp TCP-B
```

LAT-to-LAT Translation over a WAN Example

In Figure 21-5, LAT can be transported to a remote LAT device by translating the packets to TCP format and using Telnet to send them across the WAN. The configuration files for CS A and CS B follow the figure. The logical name *TS-B1* is the name given to device *TS-B*.

Figure 21-5 LAT-to-LAT Translation over a WAN



Configuration for CS A

```
interface ethernet 0
ip address 1.0.0.2 255.255.0.0
!
! enable LAT on this interface
lat enabled
!
translate lat distant-LAT tcp TS-B1
```

Configuration for CS B

```
interface ethernet 0
ip address 2.0.0.2 255.255.0.0
!
! enable LAT on this interface
lat enabled
!
translate lat TS-B1 lat LAT-B
```

Protocol Translation Session Examples

This section illustrates how to make connections for protocol translation using the one-step and two-step methods.

Using the One-Step Method for TCP-to-X.25 Host Connections

This example illustrates one-step protocol translation featuring a UNIX workstation user making a connection to a remote X.25 host named *host1* over an X.25 PDN. The protocol translator automatically converts the Telnet connection request to an X.25 connection request and transmits the request as specified in the system configuration.

- A connection is established by entering the **telnet** EXEC command at the UNIX workstation system prompt, as follows:

```
unix% telnet host1
```

Note This example implicitly assumes that the name *host1* is known to the UNIX host (obtained via DNS, IEN116, or a static table) and is mapped to the IP address used in a **translate** command.

The protocol translator accepts the Telnet connection and immediately forms an outgoing connection with remote *host1* as defined in a **translate** command in your protocol translator's active configuration file.

Next, *host1* sets several X.3 parameters, including local echo. Since the Telnet connection is already set to local echo (at the UNIX host), no changes are made on the TCP connection.

The *host1* connection prompts for a user name, then *host1* sets the X.3 parameters to cause remote echo (the same process as setting X.3 PAD parameter 2:0), and prompts for a password. The protocol translator converts this to a Telnet option request on the UNIX host, which then stops the local echo mode.

At this point the user is connected to the PAD application and the application will set the X.3 PAD parameters (although they can always be overridden by the user). When the user is finished with the connection, he enters the escape character to exit back to the host connection, then enters the appropriate command to close the connection.

The *host1* host immediately closes the X.25 connection. The protocol translator then drops the TCP connection, leaving the user back at the UNIX system prompt.

Using the Two-Step Method for TCP-to-PAD Connections

To use the two-step method, perform the following steps:

Step 1 Connect directly from a terminal or workstation to a protocol translator.

For example, you might make the following connection requests at a UNIX workstation as a first step to logging into a database called *Information Place* on an X.25 PDN:

```
unix% telnet orion
```

If the protocol translator named *orion* is accessible, it returns a login message and you enter your login name and password.

Step 2 Connect from the protocol translator to *Information Place,* which is on an X.25 host. You connect to an X.25 host using the **pad** EXEC command followed by the service address:

```
orion> pad 71330
```

Once the connection is established, the protocol translator immediately sets the PAD to single character mode with local echoing, since this is the behavior the protocol translator expects. The PAD responds with its login messages and a prompt for a password:

```
Trying 71330...Open
Welcome to the Information Place
Password:
```

Because the password should not echo on your terminal, the PAD requests remote echoing so that characters will be exchanged between the PAD and the protocol translator, but not echoed locally or displayed. After the password is verified, the PAD again requests local echoing from the protocol translator, which it does from then on.

To complete this sample session, you log off, which returns you to the protocol translator system EXEC prompt. From there, you execute the EXEC **quit** command and the protocol translator drops the network connection to the PAD.

Changing Parameters and Settings Dynamically

The following example illustrates how to make a dynamic change during a protocol translation session. In this example, you need to edit information on remote host *Information Place*. Suppose that you need to change the X.3 PAD parameters that define the editing characters from the default Delete key setting to the Ctrl-D sequence.

Step 1 Enter the escape sequence to return to the system EXEC prompt:

```
Ctrl ^ x
```

Step 2 Enter the **resume** command with the **/set** keyword and the desired X.3 parameters. X.3 parameter 16 sets the Delete function. ASCII character 4 is the Ctrl-D sequence.

```
CS> resume /set 16:4
```

The session resumes with the new settings, but now you notice that information is not being displayed correctly. You might want to set the **/debug** switch to check that your parameter setting has not been changed by the host PAD.

Step 3 Enter the escape sequence to return to the system EXEC prompt, then enter the resume command with the **/debug** switch.

```
CS> resume /debug
```

The **/debug** switch provides helpful information about the connection.

You can also set a packet dispatch character or sequence using the **terminal dispatch-character** command.

The following example shows how to set ESC (ASCII character 27) as a dispatch character:

```
CS> terminal dispatch-character 27
```

To return to the PAD connection, simply enter the following:

```
CS> resume
```