Configuring Protocol Translation and Virtual Asynchronous Devices

This chapter describes how to configure protocol translation and virtual asynchronous connections using Cisco IOS software. The tasks are described in the following sections, which also describe the process of tunneling and protocol translation, and the two-step and one-step translation methods:

The X.3 packet assembler/disassembler (PAD) parameters are described in the "X.3 PAD Parameters" appendix later in this chapter.

The protocol translation facility assumes that you understand how to use the configuration software. Before using this chapter, you should be familiar with configuring the protocols for which you want to translate: X.25, Telnet, local-area transport (LAT), TN3270, AppleTalk Remote Access (ARA), PPP, Serial Line Internet Protocol (SLIP), and XRemote.

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

Americas Headquarters:
Cisco Systems, Inc., 170 West Tasman Drive, San Jose, CA 95134-1706 USA

To identify the hardware platform or software image information associated with a feature, use the Feature Navigator on Cisco.com to search for information about the feature or refer to the software release notes for a specific release. For more information, see the "Identifying Supported Platforms" section in the "Using Cisco IOS Software" chapter.

For a complete description of the commands in this chapter, refer to *Cisco IOS Terminal Services Command Reference*. To locate documentation of other commands that appear in this chapter, use the command reference master index or search online.

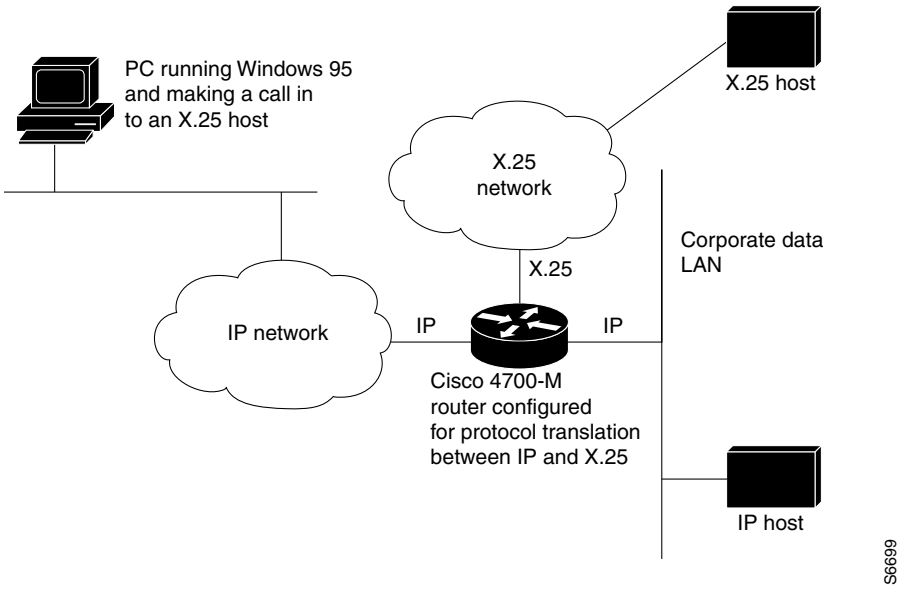
Protocol Translation Overview

Definition of Protocol Translation

The protocol translation feature provides transparent protocol translation between systems running different 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 targeted host.

Protocol translation is a resourceful facility for many business applications. For example, Figure 1 shows a remote PC dialing through an IP network and connecting to an X.25 host. The TCP packets on the PC undergo a TCP-to-X.25 protocol translation by the Cisco 4700-M router.

Figure 1 Protocol Translation Business Application

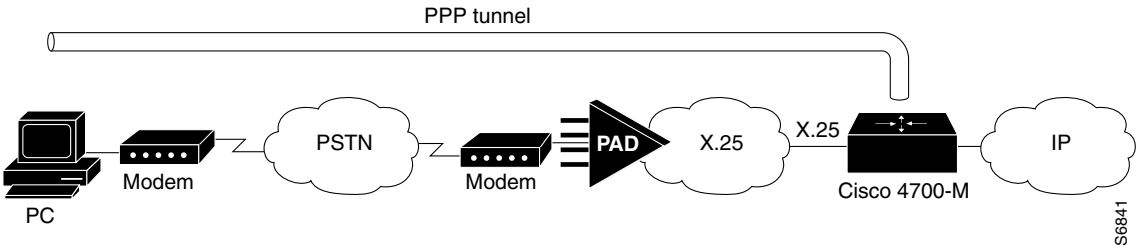


Definition of Tunneling

Unlike other protocols such as LAT, X.25, and TCP, which are actually translated when you use protocol translation, SLIP, PPP, and ARA are not translated to the destination protocol. Instead, they are carried inside a LAT, X.25, TCP, or Layer 2 Forwarding Protocol (L2F) tunnel specific to the device on the remote network. However, the protocol translation facility is used to enable tunneling of SLIP, PPP, or ARA.

Figure 2 shows a typical tunneling scenario.

Figure 2 Tunneling X.25 with PPP Across an IP Network



You can also tunnel PPP-IPX over X.25, TCP, or LAT to an Internetwork Packet Exchange (IPX) network when tunneling PPP on virtual terminal lines.

Deciding Whether to Use One-Step or Two-Step Protocol Translation

Cisco IOS software supports virtual terminal connections in both directions between the following protocols. You can configure the router to translate automatically between them. This translation method is called *one-step translation*, and is more popular than the two-step method.

- X.25 and LAT
- X.25 and Telnet sessions using the TCP
- LAT and TCP/Telnet

On outgoing connections, you can also use the one-step protocol translation facility to tunnel SLIP or PPP to IP and IPX networks, or ARA to AppleTalk networks across X.25, LAT, or IP (on outgoing connections only).

Cisco IOS software supports limited connections in both directions between the following protocols. Connecting between these protocols requires that you first connect to a router, and then to the host to which you want to connect. This translation method is called *two-step translation*, and is the less popular method.

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

One-Step Protocol Translation

Use the one-step method when network users repeatedly log in to the same remote network hosts through a router. This connection is more efficient than the two-step method and enables the device to have more knowledge of the protocols in use because the router 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 only one step to connect with the host.

When you make a one-step connection to the router, the Cisco IOS software determines the host for the connection and the protocol the host is using. It then establishes a new network connection using the protocol required by that host.

A disadvantage of the one-step protocol translation method is that the initiating computer or user does not know that two networking protocols are being used. This limitation means that parameters of 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, PPP (including IP and IPX on PPP sessions), and ARA are also supported.
- The connection features global options—You can specify additional features for the connection to allow, for example, incoming call addresses to match access list conditions or limit the number of users that can make the connection.

Refer to the "Protocol Translation Configuration Task List" section on page 5 for configuration tasks.

Two-Step Protocol Translation

Use two-step protocol translation for one-time connections or when you use the router as a general-purpose gateway between two types of networks (for example, X.25 public data network (PDN) and TCP/IP). As with the one-step method, it is recommended that you configure virtual templates for this feature.

Note 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.

Refer to the "Protocol Translation Configuration Task List" section on page 5 for configuration tasks.

Tunneling SLIP, PPP, and ARA

Unlike other protocols such as LAT, X.25, and TCP, which actually are translated when you use one-step protocol translation, SLIP, PPP, and ARA are not translated to the destination protocol. Instead, they are carried inside a LAT, X.25, or TCP tunnel specific to the device on the remote network. However, you can use the protocol translation facility to enable tunneling of SLIP, PPP, or ARA.

You can also tunnel IPX-PPP over X.25, TCP, or LAT, to an IPX network when tunneling PPP on virtual terminal lines. Refer to the "Configuring X.29 Access Lists" section on page 6 for configuration tasks.

Protocol Translation Configuration Task List

Configuring One-Step Protocol Translation

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

Command	Purpose
`Router(config)# translate protocol incoming-address`	Creates the connection specifications for one-step protocol translations.

For incoming PAD connections, the router uses a default PAD profile to set the remote X.3 PAD parameters unless a profile script is defined in the **translate** command. To override the default PAD profile that the router uses, you must create a PAD profile script using the **x29 profile** global configuration command. In the following example, *default* is the name of the default PAD profile script and *parameter:value* is the X.3 PAD parameter number and value separated by a colon.

```
x29 profile default parameter:value [parameter:value]
```

Note If the X.29 profile is named default, it is applied to all incoming X.25 PAD calls, including the calls used with protocol translation.

Configuring a Virtual Template for Two-Step Protocol Translation

If you are tunneling PPP or SLIP using two-step protocol translation with virtual interface templates, you will still use the **vty-async** command before implementing virtual templates. However, virtual asynchronous interfaces are created dynamically when a tunnel connection is established.

To create and configure a virtual interface template and apply it to a two-step protocol translation session, use the following commands in global configuration mode:

	Command	Purpose
Step 1	`Router(config)# interface virtual-template number`	Creates a virtual interface template, and enters interface configuration mode.
Step 2	`Router(config-if)# ip unnumbered ethernet 0`[1]	Assigns an IP address to the virtual interface template.
Step 3	`Router(config-if)# encapsulation {ppp | slip}`[2]	Enables encapsulation on the virtual interface template.
Step 4	`Router(config-if)# peer default ip address {dhcp | pool [pool-name-list]}`	Assigns an IP address from a pool to the device connecting to the virtual access interface (such as the PC in Figure 3).
Step 5	`Router(config-if)# exit`	Returns to global configuration mode.
Step 6	`Router(config)# vty-async`	Creates a virtual asynchronous interface.
Step 7	`Router(config)# vty-async virtual-template number`	Applies the virtual template to the virtual asynchronous interface.

1. You can also assign a specific IP address by using the **ip address** *address* command, though assigning the IP address of the Ethernet0 interface as shown is most common.
2. Virtual interface templates use PPP encapsulation by default, so you need not specify **encapsulation ppp**. However, to use SLIP encapsulation, you must explicitly specify **encapsulation slip**.

Other asynchronous configuration commands can be added to the virtual template configuration. For example, you can enter the **ppp authentication chap** command. It is recommended that you include security on your virtual interface template.

Configuring X.29 Access Lists

Cisco IOS software provides access lists to limit access to a router from certain X.25 hosts. Access lists take advantage of the message field defined by Recommendation X.29, which describes procedures for exchanging data between two PADs or between a PAD and a DTE device.

To define X.29 access lists, perform the tasks described in these sections:

- Creating an X.29 Access List, page 6 (required)
- Applying an Access List to a Virtual Line, page 7 (required)

Note When configuring protocol translation, you can specify an access list number with each **translate** command. In the case of translation sessions that result from incoming PAD connections, the corresponding X.29 access list is used.

Creating an X.29 Access List

To specify the access conditions, use the following command in global configuration mode:

Command	Purpose
`Router(config)# x29 access-list access-list-number {permit | deny} regular-expression`	Restricts incoming and outgoing connections between a particular vty (into a router) and the addresses in an access list.

An access list can contain any number of lines. The lists are processed in the order in which you type the entries. The first match causes the permit or deny condition. If an X.121 address does not match any of the entries in the access list, access will be denied.

Applying an Access List to a Virtual Line

To apply an access list to a virtual line, use the following command in line configuration mode:

Command	Purpose
`Router(config-line)# access-class number in`	Restricts incoming and outgoing connections between a particular vty (into a router) and the addresses in an access list.

The access list number is used for incoming TCP and PAD accesses. For TCP access, the access server or router using protocol translation uses the defined IP access lists. For incoming PAD connections, the same X.29 access list is used. If you want to apply access restrictions on only one of the protocols, create an access list that permits all addresses for the other protocol.

Note For an example of including an access list in a **translate** command.

Changing the Number of Supported Translation Sessions

There is a one-to-one relationship between protocol translation sessions and virtual terminal lines. For every session, you need a vty. Therefore, if you need to increase the number of protocol translation sessions, you need to increase the number of virtual terminal lines. That is, if your router has ten virtual terminal lines, you can have ten protocol translation sessions. The default number of virtual terminal lines is 5 (lines 0 through 4).

To increase the number of lines and correspondingly increase the number of protocol translation sessions, use the following commands in global configuration mode:

Command	Purpose
`Router(config)# line vty line-number`	Increases the number of virtual terminal lines.
`Router(config-line)# no line vty line-number`	Decreases the number of virtual terminal lines.

Protocol translation is a CPU-intensive task. Increasing the number of protocol translation sessions while routing is enabled can impact the available memory. The amount of memory available depends on the platform type, the amount of DRAM available, the activity of each translation 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.

Creating an X.29 Profile Script

You can create an X.29 profile script for the **translate** command to use. An X.29 profile script uses X.3 PAD parameters. When an X.25 connection is established, Cisco IOS software configured for protocol translation functions similar to an X.29 SET PARAMETER packet, which contains the parameters and values set by this command.

To create an X.29 profile script, use the following command in global configuration mode:

Command	Purpose
`Router(config)# x29 profile {default | name} parameter:value [parameter:value]`	Creates an X.29 profile script.

For incoming PAD connections, the router running protocol translation uses a default PAD profile to set the remote X.3 PAD parameters, unless a profile script is defined in the **translate** command. To override the default PAD profile that the router uses, you must create a PAD profile script and name it default using the **x29 profile** {**default** | *name*} *parameter:value* [*parameter:value*] global configuration command, where the *name* argument is the word "default" and *parameter:value* is the X.3 PAD parameter number and value separated by a colon. For more information about X.3 PAD parameters, refer to the appendix "X.3 PAD Parameters" at the end of this publication.

Note When the X.29 profile is named default, it is applied to all incoming X.25 PAD calls, including the calls used with protocol translation.

You can also create an X.29 profile script when connecting to a PAD using the **pad** [**/profile** *name*] EXEC command, which is described in *Cisco IOS Terminal Services Command Reference*.

Defining X.25 Hostnames

This section describes how to define symbolic hostnames, which means that instead of remembering a long numeric address for an X.25 host, you can refer to the X.25 host using a symbolic hostname. To define a symbolic hostname, use the following command in global configuration mode:

Command	Purpose
`Router(config)# x25 host name x.121-address [cud call-user-data]`	Defines a symbolic hostname.

Protocol Translation and Processing PAD Calls

Background Definitions and Terms

X.29 encodes the PAD Call User Data (CUD) field in the call packet to indicate that the call request signifies a PAD-to-DTE device interaction.The CUD field is 16 bytes long and can be up to 128 bytes long when the "Select" facility is applied. The first 4 bytes of the CUD field represent the protocol identifier (PID).

When a PAD calls a host DTE device, X.29 ensures that the encoding of the PID field contains a standard PAD PID "0x01000000," which informs the host that a PAD is calling. The remainder of the CUD field contains the user data that could signify a login message or a password for the host.

The **x25 map pad** interface command specifies the other end of a connection and how to interact with that host. For incoming calls, the PAD checks for a matching SOURCE address in the map entry. For outgoing calls, the PAD checks for a matching DESTINATION address in the map entry.

The **x25 map pad** commands are used to configure PAD and protocol translation accesses. They are also used to override the configuration of the interface on a per-destination basis.

The following example shows how to configure an X.25 interface to restrict incoming PAD access to a single mapped host. This example requires that both incoming and outgoing PAD accesses use the Network User Identification (NUID) to authenticate the user.

```
interface serial 0
 x25 pad-access
 x25 smap pad 219104 nuid johndoe secret
```

Accepting a PAD Call

An incoming PAD call is accepted by a Cisco router if the destination address matches the following criteria:

- A translation entry.
- The interface address.
- An alias of an interface.
- The address of the interface with trailing zeros.
- An interface subaddress.
- A NULL address.
- The address for the router set by the **x25 host** command.

When a Cisco router receives a call that requires protocol translation, the protocol translator searches the translation table for an entry with a regular expression in the X.121 address and the CUD field that matches the incoming X.121 address and the user data part of the CUD (the default PAD PID is not included).

If the PID is a nonstandard value (not equal to 0x01000000), the protocol translator searches the translation table for an entry with a regular expression in the X.121 address and the CUD field that matches the entire CUD (PID and user data).

For example, an incoming call to destination 417262510195 with a standard PAD PID of 0x01000000 and no user data will match the following translation entry:

```
translate x25 417262510195 tcp 172.31.186.54
```

An incoming call to destination 417262510195 with an unknown PID of 1234 and user data zayna will match the following translation entry:

```
translate x25 417262510195 cud 1234zayna tcp 172.31.186.54
```

An incoming call to destination 417262510195 with a standard PAD PID of 0x01000000 and user data zayna will match the following translation entry:

```
translate x25 417262510195 cud zayna tcp 172.31.186.54
```

Note Using the **translate** command, you can specify the CUD field in ASCII, octal, or hexadecimal format. You cannot enter CUD values in hexadecimal format using the **pad** command. However, you can enter the octal equivalents of CUD hexadecimal values using the following command syntax:

```
pad x121-address /cud \307\021
```

In the following example, the regular expression CUD field allows an incoming call to destination 31200100994301 with a standard PAD PID of 0x01000000 and User Data 0xD0<*whatever*> to match the following translation entry:

```
translate X25 31200100994301 cud \320.* tcp 172.20.169.11 port 13301
```

Note The PID cannot be eliminated. The entire CUD field cannot be 0. The PAD uses the PID length to determine if a PID was entered. Therefore, using the characters *""* or *\000* will be interpreted as if no PID was given.

Processing Outgoing PAD Calls Initiated by Protocol Translation

Specifying the use-map Option on Outgoing PAD and Protocol Translation Connections

Specifying the **use-map** option on the **pad** EXEC command or the **translate** global configuration command (as an outgoing protocol option) allows the optional PID, CUD, and facilities to be applied on a per-PAD connection or protocol-translation basis. If you specify the **use-map** option on the PAD connection or on the **translate** command, the DESTINATION address and (optional) PID and CUD are checked against a list of entries configured with the **x25 map pad** command.

When a match is found and the corresponding interface is available (up), the call is placed on that interface and the **x25 map** options, including facilities, are applied on the outgoing call. Otherwise, the PAD call is refused.

Note The **use-map** option is not supported on outgoing protocol translation PVCs.

For example, entering the **use-map** option on the **pad** EXEC command returns the following:

```
interface serial 1
 encapsulation x25
 x25 address 2192222
 x25 win 7
 x25 wout 7
 x25 ips 256
 x25 ops 256
 x25 map pad 77630 packetsize 1024 1024 windowsize 2 2 reverse
```

The interface in this example is configured for a window size of 7 and a packet size of 256.

The following example specifies the **use-map** option so that the outgoing PAD connection will override the interface facilities and apply a window size of 2, a packet size of 1024, and reverse charging on the outgoing PAD call:

```
pad 77630 /use-map
```

The following example specifies the **use-map** option so that a translation of the following outgoing PAD connection will cause the Call Request to be sent with a standard PAD PID and the user data to be sent in hexadecimal format:

```
! On the interface the call goes out on:
interface Serial1
 x25 map pad 417262510197 pid 0x01000000<hex for your user data>
!
translate tcp 172.21.186.54 x25 417262510197 use-map
```

The following example specifies the **use-map** options so that this outgoing PAD connection will cause the Call Request to be sent with a nonstandard PAD PID of 0x0E and user data hello:

```
! On the interface the call goes out on:
interface Serial1
 x25 map pad 417262510198 pid 0x0E cud hello
!
translate tcp 172.21.186.54 x25 417262510198 use-map
```

Applying the X.25 Route Table on Outgoing PAD and Protocol Translation Connections

When the **use-map** option is not specified on the **pad** EXEC command or the **translate** global configuration command as an outgoing protocol option, the PAD or the protocol translator locates the X.121 destination address in the X.25 route table to determine the interface on which to establish the outgoing switched virtual circuits (SVC) or permanent virtual circuits (PVCs). The destination address and optional CUD are checked against the configured list of X.25 route entries. If a matching route entry is found and the corresponding interface is operational, the call is placed on that interface. If the interface is not operational or out of available virtual circuits, the lookup for the next matching route is continued.

If the route disposition is clear, the PAD call is refused. If the route lookup does not match any valid entry, the call is placed on the first configured X.25 interface. If the default interface (that is, the first configured X.25 interface, which may or may not be up or available) is not operational or out of available virtual circuits, the PAD call is refused.

Increasing or Decreasing the Number of Virtual Terminal Lines

Because each protocol translation session uses a vty, you need to increase the number of virtual terminal lines to increase the number of protocol translation sessions. That is, if your router has ten virtual terminal lines, you can have 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, use the following commands as needed, in global configuration mode:

Command	Purpose
`Router(config)# line vty line-number`	Increases the number of virtual terminal lines.
`Router(config-line)# no line vty line-number`	Decreases the number of virtual terminal lines.

Caution Protocol translation is a CPU-intensive task. Increasing the number of protocol translation sessions while routing is enabled can impact available memory. The amount of memory available depends on the platform type, the amount of DRAM available, the activity of each translation 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.

The maximum number of protocol translation sessions for each platform can be increased to the number specified in Table 1. One virtual terminal is required for each protocol translation session.

Table 1 Maximum Number of Protocol Translation Sessions by Platform

Platform	Default Number of Virtual Terminal Lines	Total Number of Lines[1]	Maximum Virtual Terminal Lines with Translation Option
Cisco 1000 running Cisco IOS software	5	6	5
Cisco 2500 series (8 asynchronous ports)	5	200	180
Cisco 2500 series (16 asynchronous ports)	5	200	182
Cisco 2600 series	5	200	182
Cisco 3000 series	5	200	198
Cisco 3640	5	1002	872
Cisco 3620	5	1002	936
Cisco 4000 series	5	200	198
Cisco 4500 series	5	1002	1000
Cisco 4700 series	5	1002	1000
Cisco AS5200	5	200	182
Cisco AS5300	5	1002	952
Cisco 7000 series	5	120	118
Cisco 7200 series	5	1002	1000
Cisco 7000 series with RSP	5	1002	1000

1. Maximum number of virtual terminal lines = (TTYs + AUX + CON lines). Maximum number of virtual terminal lines with protocol translation option = (TTYs + AUX + CON lines).

Maintaining Virtual Interfaces

Monitoring and Maintaining a Virtual Access Interface

When a virtual interface template is applied to a protocol translation session, a virtual access interface is created dynamically. This is the only way a virtual access interface can be created. To display or clear a specific virtual access interface, use any of the following commands in user EXEC mode:

Command	Purpose
Router> **show users** [**all**]	Identifies the number associated with the virtual access interface, so you can display statistics about the interface or clear the interface.
Router> **show interfaces virtual-access** *number*	Displays the configuration of the virtual access interface.
Router> **clear interface virtual-access** *number*	Tears down the virtual access interface and frees the memory for other dial-in uses.

Displaying a Virtual Asynchronous Interface

To view information about the vty when the configuration of a virtual interface template is cloned to a vty configured as a virtual access interface for two-step protocol translation, use the following command in EXEC mode:

Command	Purpose
Router> **show line** [*line-number*]	Displays statistics about a vty.

Troubleshooting Virtual Asynchronous Interfaces

The following example shows the **debug** command output for the router redmount. It also shows the output for a specific **vty-async** interface. The **vty-async** command configures all virtual terminal lines on a router to support asynchronous protocol features.

```
Router# show debug

  PPP:
    PPP protocol negotiation debugging is on
  Asynchronous interfaces:
    Async interface framing debugging is on
    Async interface state changes debugging is on
  ROUTER1#
  ROUTER1#
  Initializing ATCP
  VTY-Async3: Set up PPP encapsulation on TTY3
  VTY-Async3: Setup PPP framing on TTY3
  VTY-Async3: Async protocol mode started for 172.22.164.1
  %LINK-3-UPDOWN: Interface VTY-Async3, changed state to up
  ppp: sending CONFREQ, type = 2 (CI_ASYNCMAP), value = A0000
  ppp: sending CONFREQ, type = 5 (CI_MAGICNUMBER), value = 91B8C7
  ppp: sending CONFREQ, type = 2 (CI_ASYNCMAP), value = A0000
  ppp: sending CONFREQ, type = 5 (CI_MAGICNUMBER), value = 91B8C7
  ROUTER1# debug 0x2
  ppp: config ACK received, type = 2 (CI_ASYNCMAP), value = A0000
  ppp: config ACK received, type = 5 (CI_MAGICNUMBER), value = 91B8C7
  ppp: config ACK received, type = 7 (CI_PCOMPRESSION)
```

```
  ppp: config ACK received, type = 8 (CI_ACCOMPRESSION)
  PPP VTY-Async3: received config for type = 0x1 (MRU) value = 0x5DC acked
  PPP VTY-Async3: received config for type = 0x2 (ASYNCMAP) value = 0x0 acked
  PPP VTY-Async3: received config for type = 0x7 (PCOMPRESSION) acked
  PPP VTY-Async3: received config for type = 0x8 (ACCOMPRESSION) acked
  ipcp: sending CONFREQ, type = 3 (CI_ADDRESS), Address = 190.0.2.255
  ppp VTY-Async3: ipcp_reqci: rcvd COMPRESSTYPE (rejected) (REJ)
  ppp VTY-Async3: Negotiate IP address: her address 10.1.1.1 (NAK with address
172.22.164.1) (NAK)
  ppp: ipcp_reqci: returning CONFREJ.
  PPP VTY-Async3: state = REQSENT fsm_rconfack(0x8021): rcvd id 0x1
  ipcp: config ACK received, type = 3 (CI_ADDRESS), Address = 172.21.213.7
  ppp VTY-Async3: Negotiate IP address: her address 10.1.1.1 (NAK with address
172.22.164.1) (NAK)
  ppp: ipcp_reqci: returning CONFNAK.
  ppp VTY-Async3: Negotiate IP address: her address 172.22.164.1 (ACK)
  ppp: ipcp_reqci: returning CONFACK.
  %LINEPROTO-5-UPDOWN: Line protocol on Interface VTY-Async3, changed state to up

  Router# show interface vty-async 3

  VTY-Async3 is up, line protocol is up
    Hardware is Virtual Async Serial
    Interface is unnumbered.  Using address of Ethernet0 (172.21.213.7)
    MTU 1500 bytes, BW 9 Kbit, DLY 100000 usec, rely 255/255, load 1/255
    Encapsulation PPP, loopback not set, keepalive set (10 sec)
    DTR is pulsed for 0 seconds on reset
    lcp state = OPEN
    ncp ccp state = NOT NEGOTIATED   ncp ipcp state = OPEN
    ncp osicp state = NOT NEGOTIATED   ncp ipxcp state = NOT NEGOTIATED
    ncp xnscp state = NOT NEGOTIATED   ncp vinescp state = NOT NEGOTIATED
    ncp deccp state = NOT NEGOTIATED   ncp bridgecp state = NOT NEGOTIATED
    ncp atalkcp state = NOT NEGOTIATED   ncp lex state = NOT NEGOTIATED
    ncp cdp state = NOT NEGOTIATED
    Last input 0:00:01, output 0:00:02, output hang never
    Last clearing of "show interface" counters never
    Input queue: 1/75/0 (size/max/drops); Total output drops: 0
    Output queue: 0/64/0 (size/threshold/drops)
       Conversations  0/1 (active/max active)
       Reserved Conversations 0/0 (allocated/max allocated)
    5 minute input rate 0 bits/sec, 0 packets/sec
    5 minute output rate 0 bits/sec, 0 packets/sec
       26 packets input, 1122 bytes, 0 no buffer
       Received 0 broadcasts, 0 runts, 0 giants
       0 input errors, 0 CRC, 0 frame, 0 overrun, 0 ignored, 0 abort
```

Monitoring Protocol Translation Connections

This section describes how to log significant virtual terminal-asynchronous authentication information such as the X.121 calling address, CUD, and IP address assigned to a virtual terminal asynchronous connection. Depending on how you configure the logging information to be displayed, you can direct this authentication information to the console, an internal buffer, or a UNIX syslog server. This authentication information can be used to associate an incoming PAD virtual terminal-asynchronous connection with an IP address.

Note By default, Cisco IOS software displays all messages to the console terminal.

To monitor protocol translation connections, perform the following tasks:

Logging vty-Asynchronous Authentication Information to the Console Terminal

To log significant vty-asynchronous authentication information to the console terminal, use the following command in global configuration mode:

Command	Purpose
`Router(config)# service pt-vty-logging`	Logs significant virtual terminal-asynchronous authentication information.

Logging vty-Asynchronous Authentication Information to a Buffer

To log significant vty-asynchronous authentication information to a buffer, use the following commands in global configuration mode as needed:

	Command	Purpose
Step 1	`Router(config)# service pt-vty-logging`	Logs significant virtual terminal-asynchronous authentication information.
Step 2	`Router(config)# logging buffered [size]`	Directs the authentication log information to a buffer.

Logging vty-Asynchronous Authentication Information to a UNIX Syslog Server

To log significant vty-asynchronous authentication information to a UNIX syslog server, use the following commands in global configuration mode as needed:

	Command	Purpose
Step 1	`Router(config)# service pt-vty-logging`	Logs significant vty-asynchronous authentication information.
Step 2	`Router(config)# logging host`	Directs the authentication log information to a UNIX syslog server.

Troubleshooting Protocol Translation

To troubleshoot your protocol translation sessions, use the following **show** and **debug** commands:

- **debug async**
- **debug pad**
- **show arap**
- **show async status**

- **show interfaces virtual-access**
- **show ip local pool**
- **show line**

Use these commands in EXEC mode. Refer to Cisco IOS command references for explanations of command output.

Virtual Template for Protocol Translation Examples

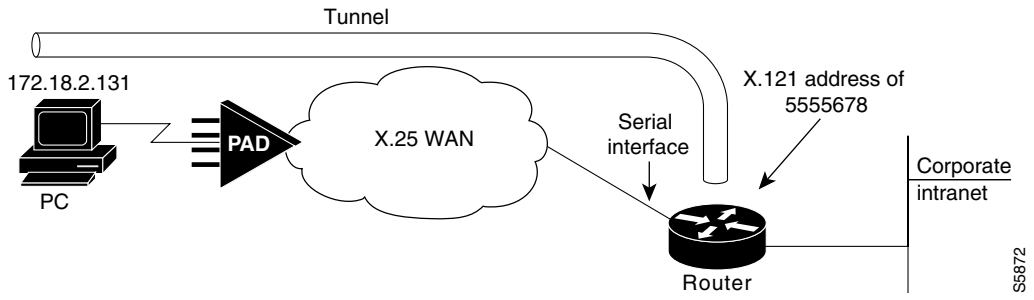
One-Step Examples

Tunnel PPP Across X.25: Example

The following example shows a virtual interface template that specifies a peer IP address of 172.18.2.131, which is the IP address of the PC in Figure 3. The virtual interface template explicitly specifies PPP encapsulation. The translation is from X.25 to PPP, which enables tunneling of PPP across an X.25 network, as shown in Figure 3.

```
interface virtual-template 1
 ip unnumbered Ethernet0
 ! Static address of 172.18.2.131 for the PC dialing in to the corporate intranet.
 peer default ip address pool group1
 ! Where the pool name is defined as ip local pool group1 172.18.35.1 172.18.35.5.
 encapsulation ppp
! X.121 address of 5555678 is the number the PAD dials to connect through the router.
translate x25 5555678 virtual-template 1
```

Figure 3 Tunneling PPP Across an X.25 Network



Tunnel SLIP Across X.25: Example

The following example uses SLIP encapsulation instead of PPP encapsulation on the virtual interface:

```
interface Virtual-Template5
 ip unnumbered Ethernet0
 encapsulation slip
 peer default ip address pool group1
 ! Where the pool name is defined as ip local pool group1 172.18.35.11 172.18.35.15.
!
translate x25 5555000 virtual-template 5
```

Tunnel PPP Across X.25 and Specify CHAP and Access List Security: Example

The following example uses PPP encapsulation on the virtual terminal interface, although it is not explicitly specified. It also uses CHAP authentication and an X.29 access list.

```
x29 access-list 1 permit ^5555
!
interface Virtual-Template1
 ip unnumbered Ethernet0
 peer default ip address pool group1
 ! Where the pool name is defined as ip local pool group1 172.18.35.21 172.18.35.25.
 ppp authentication chap
!
translate x25 5555667 virtual-template 1 access-class 1
```

Tunnel PPP with Header Compression On: Example

The following example uses TCP header compression when tunneling PPP across X.25:

```
interface Virtual-Template1
 ip unnumbered Ethernet0
 ip tcp header-compression passive
 peer default ip address pool group1
 ! Where the pool name is defined as ip local pool group1 172.18.35.31 172.18.35.35.
!
translate x25 5555676 virtual-template 1
```

Tunnel IPX-PPP Across X.25: Example

The following example shows how to tunnel IPX-PPP across the X.25 network. It creates an internal IPX network number on a loopback interface, and then assigns that loopback interface to the virtual interface template.

```
ipx routing 0000.0c07.b509
!
interface loopback0
 ipx network 544
 ipx sap-interval 2000
!
interface Virtual-Template1
 ip unnumbered Ethernet0
 ipx ppp-client Loopback0
 peer default ip address pool group1
 ! Where the pool name is defined as ip local pool group1 172.18.35.41 172.18.35.45.
!
translate x25 5555766 virtual-template 1
```

Two-Step Examples

Two-Step Tunneling of PPP with Dynamic Routing and Header Compression: Example

The following example uses the default PPP encapsulation on the virtual template.

```
vty-async
vty-async virtual-template 1
vty-async dynamic-routing
vty-async header-compression
!
interface Virtual-Template1
 ip unnumbered Ethernet0
 no peer default ip address
```

After users connect to the router (in this example, named waffler), they invoke the **ppp** command to complete the two-step connection:

```
Router> ppp /routing /compressed 172.16.2.31
Entering PPP routing mode.
Async interface address is unnumbered (Ethernet0)
Your IP address is 172.16.2.31. MTU is 1500 bytes
```

Two-Step Tunneling of PPP with Dynamic Routing, TACACS, and CHAP: Example

The virtual template interface in the following example uses the default encapsulation of PPP and applies CHAP authentication with TACACS+:

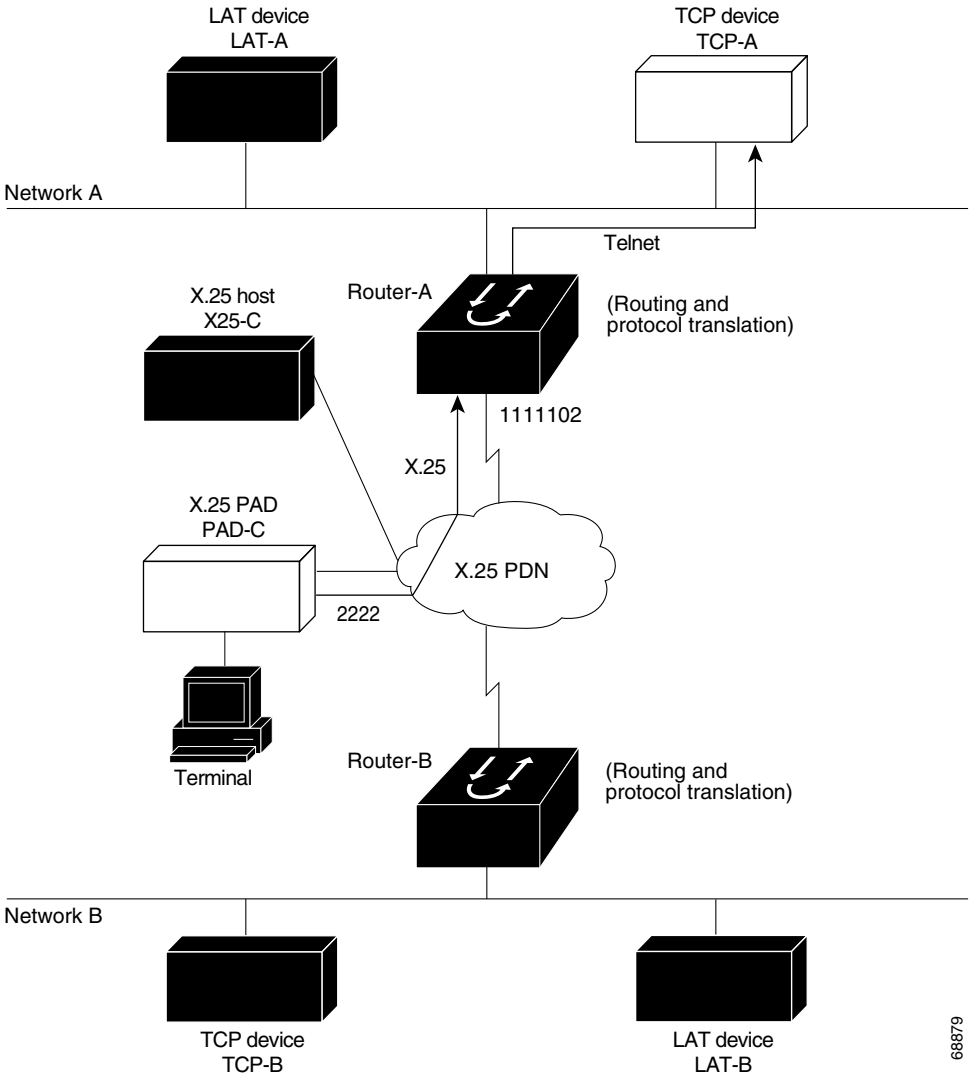
```
aaa authentication ppp default tacacs+
!
vty-async
vty-async dynamic-routing
vty-async virtual-template 1
!
interface Ethernet0
 ip address 10.11.12.2 255.255.255.0
!
interface Virtual-Template1
 ip unnumbered Ethernet0
 no peer default ip address
 ppp authentication chap
```

Protocol Translation Application Examples

X.25 PAD-to-TCP Configuration: Example

Making a translated connection from an X.25 PAD to a TCP device is analogous to the preceding X.25 PAD-to-LAT example. (See Figure 4.) Instead of translating to LAT, the configuration for Router-A includes a statement to translate to TCP (Telnet). Note that a router with the protocol translation software option can include statements supporting both translations (X.25 PAD to LAT and X.25 PAD to TCP). Different users on the same PAD can communicate with X.25, LAT, or TCP devices.

Figure 4 X.25 PAD-to-TCP Translation



The following example shows how to use the **translate** global configuration command to translate from an X.25 PAD to a TCP device on Network A. It is applied to Router-A.

```
! Set up translation.
translate x25 2222 tcp TCP-A
```

Protocol Translation Session Examples

One-Step Method for TCP-to-X.25 Host Connections: Example

This example demonstrates 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 router automatically converts the Telnet connection request to an X.25 connection request and sends the request as specified in the system configuration.

A connection is established when you enter 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 router accepts the Telnet connection and immediately forms an outgoing connection with remote host1 as defined in a **translate** command.

Next, host1 sets several X.3 parameters, including local echo. Because 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 username, 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. Cisco IOS software converts this request 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 finished with the connection, the user escapes back to the host connection, and then enters the appropriate command to close the connection.

The host named host1 immediately closes the X.25 connection. The Cisco IOS software then drops the TCP connection, leaving the user back at the UNIX system prompt.

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

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

Step 1 Connect directly from a terminal or workstation to a router.

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

```
unix% telnet orion
```

If the router named orion is accessible, it returns a login message, and you can enter your login name and password.

Step 2 Connect from the router 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:

```
Router> pad 71330
```

Once the connection is established, the router immediately sets the PAD to single-character mode with local echoing, because these are the settings the router 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 router, but not echoed locally or displayed. After the password is verified, the PAD again requests local echoing from the router, which it does from then on.

To complete this sample session, log out to return to the router system EXEC prompt and enter the EXEC **quit** command; the router drops the network connection to the PAD.

Two-Step Protocol Translation for TCP-to-PAD Connections: Example

The following example shows a connection from a local UNIX host named host1 to a router named router1 as the first step in a two-step translation process:

```
host1% telnet Router1
```

The following sample session shows a connection from Router1 to a host named ibm3278 as the second step in a two-step translation process:

```
Router1> tn3270 ibm3278
ibm3278%
```

Next, connect directly from a terminal or workstation on a TCP/IP network to a router, and then to a database named Information Place on an X.25 packet data network. The database has a service address of 71330.

To complete the two-step translation connection, perform the following steps:

Step 1 Make the following connection requests at a UNIX workstation as a first step to logging in to the database Information Place:

```
unix% telnet router1
```

If the router named router1 is accessible, it returns a login message and you can enter your login name and password.

Step 2 Connect from the router to the 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:

```
Router1> pad 71330
```

Once the connection is established, the router immediately sets the PAD to single-character mode with local echoing, because these are the settings that the router 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 router, but not echoed locally or displayed. After the password is verified, the PAD again requests local echoing from the router.

Step 3 Complete the session by logging out, which returns you to the router system EXEC prompt.

Step 4 Enter the **quit** EXEC command; the router drops the network connection to the PAD.

Changing Parameters and Settings Dynamically: Example

The following sample session shows how to make a dynamic change during a protocol translation session. In this sample, you will edit information on the remote host named Information Place. To change the X.3 PAD parameters that define the editing characters from the default Delete key setting to the Ctrl-D sequence, perform the following steps:

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.

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

The session resumes with the new settings, but the information is not displayed correctly. You may 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, and then enter the **resume** command with the **/debug** switch.

```
Router> 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:

```
Router> terminal dispatch-character 27
```

To return to the PAD connection, enter the **resume** command:

```
Router> resume
```

Monitoring Protocol Translation Connections: Example

The following example shows how to log significant virtual terminal-asynchronous authentication information, such as the X.121 calling address, CUD, and IP address assigned to a virtual terminal-asynchronous connection, to a UNIX syslog server named alice:

```
service pt-vty-logging
logging alice
```

Two-Step Protocol Translation for Virtual Terminal Asynchronous Interfaces: Example

Caution The following example shows how to configure the **vty-async** command for PPP over X.25 using the router named redmount:

```
hostname redmount

ip address-pool local
x25 routing
vty-async                   <------ two-step translation
vty-async dynamic-routing   <------ optional
vty-async mtu 245           <------ optional

interface Ethernet0
 ip address 172.31.113.7 255.255.255.0
 no mop enabled

interface Serial0
 no ip address
 encapsulation x25
 x25 address 9876543210

router rip
 network 172.31.213.0
 network 172.22.164.0

ip domain-name cisco.com
ip name-server 172.31.213.2
ip name-server 172.31.213.4
ip local pool default 172.22.164.1 172.28.164.254
x25 route 9876543211 alias serial 0
x25 route 9876543212 alias serial 0

line con 0
 exec-timeout 0 0
line aux 0
 transport input all
line vty 0 1                <------ used for remote access to the router
 rotary 2
line vty 2 64               <------ used for ppp over x25
 rotary 1
 autocommand ppp default
```