

Configuring X.25 and LAPB

This chapter describes how to configure connections through Link Access Procedure, Balanced (LAPB) connections and X.25 networks. LAPB procedures are presented first for those users who only want to configure a simple, reliable serial encapsulation method. This chapter also describes how to configure Defense Data Network (DDN) X.25 and the Blacker Front End (BFE), and how to create X.29 access lists.

For a complete description of the commands mentioned in this chapter, refer to the “X.25 and LAPB Commands” chapter in the *Wide-Area Networking Command Reference*. To locate documentation of other commands that appear in this chapter, use the command reference master index or search online.

To configure PAD access, refer to the “Configuring the Cisco PAD Facility for X.25 Connections” chapter in the *Dial Solutions Configuration Guide*.

To translate between an X.25 packet assembler/disassembler (PAD) connection and another protocol, refer to the “Protocol Translation Commands” chapter in the *Dial Solutions Command Reference*.

To configure X.25 over ISDN, refer to the “Configuring ISDN” chapter in the *Dial Solutions Configuration Guide*. For a complete description of the commands, refer to the “ISDN Commands” chapter of the *Dial Solutions Command Reference*.

LAPB Configuration Task List

You can use only LAPB as a serial encapsulation method if you have a private serial line. You must use one of the X.25 packet-level encapsulations when attaching to an X.25 network.

The LAPB standards distinguish between two types of hosts: data terminal equipment (DTE), and data circuit-terminating equipment (DCE). At Level 2, or the data link layer in the OSI model, LAPB allows for orderly and reliable exchange of data between a DTE and a DCE. A router using LAPB encapsulation can act as a DTE or DCE device at the protocol level, which is distinct from the hardware DTE or DCE identity.

Using LAPB under noisy conditions can result in greater throughput than High-Level Data Link Control (HDLC) encapsulation. When LAPB detects a missing frame, the router retransmits the frame instead of waiting for the higher layers to recover the lost information. This behavior is good only if the host timers are relatively slow. In the case of quickly expiring host timers, however, you will discover that LAPB is spending much of its time transmitting host retransmissions. If the line is not noisy, the lower overhead of HDLC encapsulation is more efficient than LAPB. When you are using long-delay satellite links, for example, the lock-step behavior of LAPB makes HDLC encapsulation the better choice.

To configure LAPB, complete the tasks in the following sections. The tasks in the first section are required; the remaining are optional.

- Configure a LAPB Datagram Transport
- Modify LAPB Protocol Parameters
- Configure Priority and Custom Queuing for LAPB
- Configure Transparent Bridging over Multiprotocol LAPB
- Monitor and Maintain LAPB and X.25

Examples of LAPB configurations are given at the end of this chapter.

Configure a LAPB Datagram Transport

To set the appropriate LAPB encapsulation to run datagrams over a serial interface, perform the following task in global configuration mode. One end of the link must be DTE, and the other must be DCE.

Task	Command
Specify a serial interface.	interface serial <i>number</i>

Because the default serial encapsulation is HDLC, you must explicitly configure a LAPB encapsulation method.

Note It is recommended that you shut down an interface before changing the encapsulation.

To select an encapsulation and the protocol if using a single protocol, or to select the multiple protocol operation, perform one or more of the following tasks in interface configuration mode:

Task	Command
Enable encapsulation of a single protocol on the line using DCE operation.	encapsulation lpb dce [<i>protocol</i>] ¹
Enable encapsulation of a single protocol on the line using DTE operation.	encapsulation lpb dte [<i>protocol</i>] ¹
Enable use of multiple protocols on the line using DCE operation.	encapsulation lpb dce multi
Enable use of multiple protocols on the line using DTE operation.	encapsulation lpb dte multi ^{2, 3}

1. Single protocol LAPB defaults to IP encapsulation.
2. Multiprotocol LAPB does not support source-route bridging or TCP/IP header compression, but does support transparent bridging.
3. A multiprotocol LAPB encapsulation supports all of the protocols available to a single protocol LAPB encapsulation plus transparent bridging.

For an example of configuring LAPB operation, see the section “Typical LAPB Configuration Example” later in this chapter.

Configure Compression of LAPB Data

You can configure point-to-point software compression on serial interfaces that use a LAPB or multi-LAPB encapsulation. Compression reduces the size of a LAPB or multi-LAPB frame via lossless data compression.

Compression is performed in software and can significantly affect system performance. Cisco recommends that you disable compression if the router CPU load exceeds 65 percent. To display the CPU load, use the **show process cpu EXEC** command.

Predictor compression is recommended when the bottleneck is caused by the load on the router or access server; Stacker compression is recommended when the bottleneck is the result of line bandwidth. Compression is not recommended if the majority of your traffic is already compressed files. Compression is also not recommended for line speeds greater than T1; the added processing time slows performance on fast lines.

To configure compression over LAPB, perform the following tasks in interface configuration mode:

Task	Command
Step 1 Enable encapsulation of a single protocol on the serial line.	encapsulation lapb <i>[protocol]</i>
Step 2 Enable compression.	compress <i>[predictor stac]</i>

To configure compression over multi-LAPB, perform the following tasks in interface configuration mode:

Task	Command
Step 1 Enable encapsulation of multiple protocols on the serial line.	encapsulation lapb multi
Step 2 Enable compression.	compress <i>[predictor stac]</i>

When you are using compression, adjust the maximum transmission unit (MTU) for the serial interface and the LAPB N1 parameter as in the following example, to avoid informational diagnostics regarding excessive MTU or N1 sizes:

```
interface serial 0
 encapsulation lapb
 compress predictor
 mtu 1509
 lapb n1 12072
```

For information about configuring X.25 TCP/IP header compression and X.25 payload compression, see the “Set X.25 TCP/IP Header Compression” and “Configure X.25 Payload Compression” sections later in this chapter.

Modify LAPB Protocol Parameters

X.25 Level 2, or LAPB, operates at the data link layer of the OSI reference model. LAPB specifies methods for exchanging data (in units called *frames*), detecting out-of-sequence or missing frames, retransmitting frames, and acknowledging frames. Several protocol parameters can be modified to

change LAPB protocol performance on a particular link. Because X.25 operates the Packet Level Protocol (PLP) on top of the LAPB protocol, these tasks apply to both X.25 links and LAPB links. The parameters and their default values are summarized in Table 2.

Table 2 LAPB Parameters

Task (LAPB Parameter)	Command	Values or Ranges	Default
Set the modulo.	lapb modulo <i>modulus</i>	8 or 128	8
Set the window size (K).	lapb k <i>window-size</i>	1- (modulo minus 1) frames	7
Set the maximum bits per frame (N1).	lapb n1 <i>bits</i>	Bits (must be a multiple of 8)	Based on hardware MTU and protocol overhead
Set the count for sending frames (N2).	lapb n2 <i>tries</i>	1–255 tries	20
Set the retransmission timer (T1).	lapb t1 <i>milliseconds</i>	1–64000 milliseconds	3000
Set the hardware outage period.	lapb interface-outage <i>milliseconds</i>		0 (disabled)
Set the idle link period (T4).	lapb t4 <i>seconds</i>		0 (disabled)

- **LAPB Modulo and LAPB K**—The LAPB modulo determines the operating mode. Modulo 8 (basic mode) is widely available, because it is required for all standard LAPB implementations and is sufficient for most links. Modulo 128 (extended mode) can achieve greater throughput on high-speed links that have a low error rate (some satellite links, for example) by increasing the number of frames that can be transmitted before waiting for acknowledgment (as configured by the LAPB window parameter, k). By its design, LAPB’s k parameter can be at most one less than the operating modulo. Modulo 8 links can typically send seven frames before an acknowledgment must be received; modulo 128 links can set k to a value as large as 127. By default, LAPB links use the basic mode with a window of 7.
- **LAPB N1**—When connecting to an X.25 network, use the N1 parameter value set by the network administrator. This value is the maximum number of bits in a LAPB frame, which determines the maximum size of an X.25 packet. When you are using LAPB over leased lines, the N1 parameter should be eight times the hardware maximum transmission unit (MTU) size plus any protocol overhead.

The LAPB N1 range is dynamically calculated by the Cisco IOS software whenever an MTU change, an L2/L3 modulo change, or a compression change occurs on a LAPB interface.



Caution The LAPB N1 parameter provides little benefit beyond the interface MTU, and can easily cause link failures if misconfigured. Cisco recommends that this parameter be left at its default value.

- **LAPB N2**—The transmit counter (N2) is the number of unsuccessful transmit attempts that are made before the link is declared down.
- **LAPB T1**—The retransmission timer (T1) determines how long a transmitted frame can remain unacknowledged before the Cisco IOS software polls for an acknowledgment. For X.25 networks, the retransmission timer setting should match that of the network.

For leased-line circuits, the T1 timer setting is critical because the design of LAPB assumes that a frame has been lost if it is not acknowledged within period T1. The timer setting must be large enough to permit a maximum-sized frame to complete one round trip on the link. If the timer setting is too small, the software will poll before the acknowledgment frame can return, which may result in duplicated frames and severe protocol problems. If the timer setting is too large, the software waits longer than necessary before requesting an acknowledgment, which reduces bandwidth.

- LAPB interface outage—Another LAPB timer function allows brief hardware failures while the protocol is up, without requiring a protocol reset. When a brief hardware outage occurs, the link will continue uninterrupted if the outage is corrected before the specified hardware outage period expires.
- LAPB T4—The LAPB standards define a timer to detect unsignaled link failures (T4). The T4 timer is reset every time a frame is received from the partner on the link. If the T4 timer expires, a Receiver Ready frame with the Poll bit set is sent to the partner, which is required to respond. If the partner does not respond, the standard polling mechanism is used to determine whether the link is down. The period of T4 must be greater than the period of T1.

For an example of configuring the LAPB T1 timer, see the section “Typical LAPB Configuration Example” later in this chapter.

Configure Priority and Custom Queuing for LAPB

Cisco priority queuing and custom queuing are available for LAPB to allow you to improve a link’s responsiveness to a given type of traffic by specifying the handling of that type of traffic for transmission on the link.

Priority queuing is a mechanism that classifies packets based on certain criteria and then assigns the packets to one of four output queues, with high, medium, normal, or low priority. Custom queuing similarly classifies packets and assigns them to one of 10 output queues, and controls the percentage of an interface’s available bandwidth that is used for a queue.

For example, you can use priority queuing to ensure that all Telnet traffic is processed promptly and that Simple Mail Transfer Protocol (SMTP) traffic is sent only when there is no other traffic to send. Priority queuing in this example can starve the non-Telnet traffic; custom queuing can be used instead to ensure that some traffic of all categories is sent.

Both priority queuing and custom queuing can be defined, but only one method can be assigned to a given interface.

To configure priority and custom queuing for LAPB, perform the following tasks:

- 1 Perform the standard priority and custom queuing tasks *except* the task of assigning a priority or custom group to the interface, as described in the “Performing Basic System Management” chapter in the *Configuration Fundamentals Configuration Guide*.
- 2 Perform the standard LAPB encapsulation tasks, as specified in the “Configure a LAPB Datagram Transport” section of this chapter.
- 3 Assign either a priority group or a custom queue to the interface, as described in the “Performing Basic System Management” chapter in the *Configuration Fundamentals Configuration Guide*.

Note The `lapb hold-queue` command is no longer supported, but the same functionality is provided by the standard queue control command `hold-queue size out`.

Configure Transparent Bridging over Multiprotocol LAPB

To configure transparent bridging over multiprotocol LAPB, perform the following tasks beginning in global configuration mode:

Task	Command
Specify the serial interface, and enter interface configuration mode.	interface serial <i>number</i>
Assign no IP address to the interface.	no ip address
Configure multiprotocol LAPB encapsulation.	encapsulation lapb multi
Assign the interface to a bridge group.	bridge-group <i>bridge-group</i>
Define the type of spanning tree protocol.	bridge <i>bridge-group</i> protocol { ieee dec }

Note This feature requires use of the **encapsulation lapb multi** command. You cannot use the **encapsulation lapb protocol** command with a **bridge** keyword to configure this feature.

For an example of configuring transparent bridging over multiprotocol LAPB, see the section “Transparent Bridging for Multiprotocol LAPB Encapsulation Example” later in this chapter.

X.25 Configuration Task List

To configure X.25, complete the tasks in one or more of the following sections, depending upon the X.25 application or task required for your network. The interface, datagram transport, and routing tasks are divided into sections based generally on how common the feature is and how often it is used. Those features and parameters that are relatively uncommon are found in the “Additional” sections. LAPB frame parameters can be modified to optimize X.25 operation, as described earlier in this chapter.

- Configure an X.25 Interface
- Configure Additional X.25 Interface Parameters
- Configure an X.25 Datagram Transport
- Configure Additional X.25 Datagram Transport Features
- Configure Priority Queuing or Custom Queuing for X.25
- Configure X.25 Routing
- Configure Additional X.25 Routing Features
- Configure CMNS Routing
- Configure DDN or BFE X.25
- Create X.29 Access Lists
- Create an X.29 Profile Script
- Monitor and Maintain LAPB and X.25

All these features can coexist on an X.25 interface.

Default parameters are provided for X.25 operation; however, you can change the settings to meet the needs of your X.25 network or as defined by your X.25 service supplier. Cisco also provides additional configuration settings to optimize your X.25 usage.

Note If you connect a router to an X.25 network, use the parameters set by your network administrator for the connection; these parameters will typically be those described in the “Configure an X.25 Interface” and “Modify LAPB Protocol Parameters” sections. Also, note that the X.25 Level 2 parameters described in the “Modify LAPB Protocol Parameters” section affect X.25 Level 3 operations.

See the end of this chapter for examples of configuring X.25.

Configure an X.25 Interface

To configure an X.25 interface, perform the tasks in the following sections:

- Set the X.25 Mode
- Set the Virtual Circuit Ranges
- Set the Packet Numbering Modulo
- Set the X.121 Address
- Set the Default Flow Control Values

These tasks describe the parameters that are essential for correct X.25 behavior. The first task is required. The others might be required or optional, depending on what the router is expected to do with the X.25 attachment.

You can also configure other, less common parameters, as specified in the “Configure Additional X.25 Interface Parameters” section.

Set the X.25 Mode

A router using X.25 Level 3 encapsulation can act as a DTE or DCE protocol device (according to the needs of your X.25 service supplier), can use Defense Data Network (DDN) or Blacker Front End (BFE) encapsulation, or can use the Internet Engineering Task Force (IETF) standard encapsulation, as specified by RFC 1356.

Because the default serial encapsulation is HDLC, you must explicitly configure an X.25 encapsulation method.

Note It is recommended that you shut down an interface before changing the encapsulation.

To configure the mode of operation and one of these encapsulation types for a specified interface, perform the following task in interface configuration mode:

Task	Command
Set the X.25 mode of operation.	<code>encapsulation x25 [dte dce] [[ddn bfe] [ietf]]</code>

Typically a public data network (PDN) will require attachment as a DTE. (This requirement is distinct from the hardware interface DTE or DCE identity.)

The default mode of operation is DTE, and the default encapsulation method is Cisco's pre-IETF method. If either DDN or BFE operation is needed, it must be explicitly configured.

For an example of configuring X.25 DTE operation, see the section "Typical X.25 Configuration Example" later in this chapter.

Set the Virtual Circuit Ranges

The X.25 protocol maintains multiple connections over one physical link between a DTE and a DCE. These connections are called *virtual circuits* (VCs) or *logical channels* (LCs). X.25 can maintain up to 4095 virtual circuits numbered 1 through 4095. You identify an individual virtual circuit by giving its logical channel identifier (LCI) or virtual circuit number (VCN). Many documents use the terms *virtual circuit* and *LC*, *VCN*, *LCN*, and *LCI* interchangeably. Each of these terms refers to the virtual circuit number.

An important part of X.25 operation is the range of virtual circuit numbers. Virtual circuit numbers are broken into four ranges (listed here in numerically increasing order):

- 1 Permanent virtual circuits (PVCs)
- 2 Incoming-only circuits
- 3 Two-way circuits
- 4 Outgoing-only circuits

The incoming-only, two-way, and outgoing-only ranges define the virtual circuit numbers over which a switched virtual circuit (SVC) can be established by the placement of an X.25 call, much like a telephone network establishes a switched voice circuit when a call is placed.

The rules about DCE and DTE devices initiating calls are as follows:

- Only the DCE device can initiate a call in the incoming-only range.
- Only the DTE device can initiate a call in the outgoing-only range.
- Both the DCE device and the DTE device can initiate a call in the two-way range.

(The ITU-T Recommendation X.25 defines "incoming" and "outgoing" in relation to the DTE or DCE interface role; Cisco's documentation uses the more intuitive sense. Unless the ITU-T sense is explicitly referenced, a call received from the interface is an *incoming call* and a call sent out the interface is an *outgoing call*.)

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

There is no difference in the operation of the SVCs in the different ranges except the restrictions on which device can initiate a call. These ranges can be used to prevent one side from monopolizing the virtual circuits, which can be useful for X.25 interfaces with a small total number of SVCs available.

Six X.25 parameters define the upper and lower limit of each of the three SVC ranges. A PVC must be assigned a number less than the numbers assigned to the SVC ranges. An SVC range is not allowed to overlap another range.

Note Because the X.25 protocol requires the DTE and DCE to have identical virtual circuit ranges, changes you make to the virtual circuit range limits when the interface is up are held until the X.25 protocol restarts the packet service.

To configure X.25 virtual circuit ranges, complete the following tasks in interface configuration mode as appropriate for your configuration:

Task	Command	Default
Set the lowest incoming-only circuit number.	x25 lic <i>circuit-number</i>	0
Set the highest incoming-only circuit number.	x25 hic <i>circuit-number</i>	0
Set the lowest two-way circuit number.	x25 ltc <i>circuit-number</i>	1
Set the highest two-way circuit number.	x25 htc <i>circuit-number</i>	1024—for X.25 4095—for CMNS
Set the lowest outgoing-only circuit number.	x25 loc <i>circuit-number</i>	0
Set the highest outgoing-only circuit number.	x25 hoc <i>circuit-number</i>	0

Each of these parameters can range from 1 to 4095, inclusive. Note that the values for these parameters must be the same on both ends of an X.25 link. For connection to a public data network (PDN), these values must be set to the values assigned by the network. An SVC range is unused if its lower and upper limits are set to 0; other than this use for marking unused ranges, virtual circuit 0 is not available.

For an example of configuring virtual circuit ranges, see the section “Virtual Circuit Ranges Example” later in this chapter.

Set the Packet Numbering Modulo

Cisco’s implementation of X.25 supports both modulo 8 and modulo 128 packet sequence numbering; module 8 is the default.

To set the packet numbering modulo, perform the following task in interface configuration mode:

Task	Command
Set the packet numbering modulo (the default is 8).	x25 modulo {8 128}

Note Because the X.25 protocol requires the DTE and DCE to have identical modulos, changes you make to the modulo when the interface is up are held until the X.25 protocol restarts the packet service.

The X.25 modulo and the LAPB modulo are distinct and serve different purposes. LAPB modulo 128 (or extended mode) can be used to achieve higher throughput across the DTE or DCE interface; it affects only the local point of attachment. X.25 Packet-Level Protocol (PLP) modulo 128 can be used to achieve higher end-to-end throughput for virtual circuits by allowing more data packets to be in transit through the X.25 network.

Set the X.121 Address

If your router does not originate or terminate calls but only participates in X.25 switching, this task is optional. However, if the router is attached to a PDN, you must set the interface X.121 address assigned by the X.25 network service provider. Interfaces that use the DDN or BFE mode will have an X.121 address generated from the interface IP address; for correct DDN or BFE operation, any such X.121 address must not be modified.

To set the X.121 address, perform the following task in interface configuration mode:

Task	Command
Set the X.121 address.	x25 address <i>x121-address</i>

For an example of configuring the X.25 interface address, see the section “Typical X.25 Configuration Example” later in this chapter.

Set the Default Flow Control Values

Setting correct default flow control parameters of window size and packet size is essential for correct operation of the link because X.25 is a strongly flow controlled protocol. However, it is easy to overlook this task because many networks use standard default values. Mismatched default flow control values will cause X.25 local procedure errors, evidenced by Clear and Reset events.

To configure flow control parameters, complete the tasks in the following sections. These tasks are optional if your X.25 attachment uses the standard default values for maximum packet sizes (128 bytes incoming and outgoing) and window sizes (2 packets incoming and outgoing).

- Set Default Window Sizes
- Set Default Packet Sizes

Note Because the X.25 protocol requires the DTE and DCE to have identical default maximum packet sizes and default window sizes, changes made to the window and packet sizes when the interface is up are held until the X.25 protocol restarts the packet service.

Set Default Window Sizes

X.25 networks have a default input and output window size (the default is 2) that is defined by your network administrator. You must set the Cisco IOS software default input and output window sizes to match those of the network. These defaults are the values that an SVC takes on if it is set up without explicitly negotiating its window sizes. Any PVC also uses these default values unless different values are configured.

To set the default window sizes, perform the following tasks in interface configuration mode:

Task	Command
Set the default virtual circuit receive window size.	x25 win <i>packets</i>
Set the default virtual circuit transmit window size.	x25 wout <i>packets</i>

For an example of setting the default window sizes, see the sections “Typical X.25 Configuration Example” and “DDN X.25 Configuration Example” later in this chapter.

Set Default Packet Sizes

X.25 networks have a default maximum input and output packet size (the default is 128) that is defined by your network administrator. You must set the Cisco IOS software default input and output maximum packet sizes to match those of the network. These defaults are the values that an SVC takes on if it is set up without explicitly negotiating its maximum packet sizes. Any PVC also uses these default values unless different values are configured.

To set the default input and output maximum packet sizes, perform the following tasks in interface configuration mode:

Task	Command
Set the default input maximum packet size.	x25 ips <i>bytes</i>
Set the default output maximum packet size.	x25 ops <i>bytes</i>

To send a packet larger than the agreed-on X.25 packet size over an X.25 virtual circuit, the Cisco IOS software must break the packet into two or more X.25 packets with the M-bit (“more data” bit) set. The receiving device collects all packets in the M-bit sequence and reassembles them into the original packet.

It is possible to define default packet sizes that cannot be supported by the lower layer (see the LAPB N1 parameter). However, the router will negotiate lower maximum packet sizes for all SVCs so the agreed-on sizes can be carried. The Cisco IOS software will also refuse a PVC configuration if the resulting maximum packet sizes cannot be supported by the lower layer.

For an example of setting the default maximum packet sizes, see the sections “Typical X.25 Configuration Example” and “DDN X.25 Configuration Example” later in this chapter.

Configure Additional X.25 Interface Parameters

Some X.25 applications have less common or special needs. Several X.25 parameters are available to modify the X.25 protocol behavior for these applications.

To configure less common X.25 interface parameters for these special needs, perform the tasks in the following sections, as needed:

- Configure the X.25 Level 3 Timers
- Configure X.25 Addresses
- Establish a Default Virtual Circuit Protocol
- Disable Packet-Level Protocol (PLP) Restarts

Configure the X.25 Level 3 Timers

The X.25 Level 3 event timers determine how long the Cisco IOS software waits for acknowledgment of control packets. You can set these timers independently. Only those timers that apply to the interface are configurable. (A DTE interface does not have the T1x timers, and a DCE interface does not have the T2x timers.)

To set the event timers, perform any of the following tasks in interface configuration mode:

Task	Command
Set DTE T20 Restart Request timeout.	x25 t20 <i>seconds</i>
Set DCE T10 Restart Indication timeout.	x25 t10 <i>seconds</i>

Task	Command
Set DTE T21 Call Request timeout.	x25 t21 <i>seconds</i>
Set DCE T11 Incoming Call timeout.	x25 t11 <i>seconds</i>
Set DTE T22 Reset Request timeout.	x25 t22 <i>seconds</i>
Set DCE T12 Reset Indication timeout.	x25 t12 <i>seconds</i>
Set DTE T23 Clear Request timeout.	x25 t23 <i>seconds</i>
Set DCE T13 Clear Indication timeout.	x25 t13 <i>seconds</i>

For an example of setting the event timers, see the section “DDN X.25 Configuration Example” later in this chapter.

Configure X.25 Addresses

When establishing SVCs, X.25 uses addresses in the form defined by the ITU-T *Recommendation X.121* (or simply an “X.121 address”). An X.121 address has from zero to 15 digits. Because of the importance of addressing to call setup, several interface addressing features are available for X.25.

To configure X.25 addresses, perform the tasks in the following sections:

- Understand Normal X.25 Addressing
- Understand X.25 Subaddresses
- Configure an Interface Alias Address
- Suppress or Replace the Calling Address
- Suppress the Called Address

Understand Normal X.25 Addressing

An X.25 interface’s X.121 address is used when it is the source or destination of an X.25 call. The X.25 call setup procedure identifies both the calling (source) and the called (destination) X.121 addresses. When an interface is the source of a call, it encodes the interface X.121 address as the source address. An interface determines that it is the destination of a received call if the destination address matches the interface’s address.

Cisco’s X.25 software can also route X.25 calls, which involves placing and accepting calls, but the router is neither the source nor the destination for these calls. Routing X.25 does not modify the source or destination addresses, thus preserving the addresses specified by the source host. Routed (switched) X.25 simply connects two logical X.25 channels to complete an X.25 virtual circuit. An X.25 virtual circuit, then, is a connection between two hosts (the source host and the destination host) that is switched between zero or more routed X.25 links.

The null X.121 address (the X.121 address that has zero digits) is a special case. The router acts as the destination host for any call it receives that has the null destination address.

Understand X.25 Subaddresses

A subaddress is an X.121 address that matches the digits defined for the interface’s X.121 address, but has one or more additional digits after the base address. X.25 acts as the destination host for an incoming packet assembler/disassembler (PAD) call with a destination that is a subaddress of the interface’s address; the trailing digits specify which line a PAD connection is requesting. This feature

is described in the “Configuring Protocol Translation and Virtual Asynchronous Devices” chapter in the *Dial Solutions Configuration Guide*. Other calls that use a subaddress can be accepted if the trailing digit or digits are zeros; otherwise, the router will not act as the call’s destination host.

Configure an Interface Alias Address

You can supply alias X.121 addresses for an interface. This allows the interface to act as the destination host for calls having a destination address that is neither the interface’s address, an allowed subaddress of the interface, nor the null address.

Local processing (for example, IP encapsulation) can be performed only for incoming calls whose destination X.121 address matches the serial interface or alias of the interface.

To configure an alias, perform the following task in configuration mode:

Task	Command
Supply an alias X.121 address for the interface.	x25 alias <i>x121-address-pattern</i> [cmd pattern]

Suppress or Replace the Calling Address

Some attachments require that no calling (source) address be presented in outgoing calls; this requirement is called *suppressing the calling address*.

When attached to a PDN, X.25 may need to ensure that outgoing calls only use the assigned X.121 address for the calling (source) address. Routed X.25 normally uses the original source address. Although individual X.25 route configurations can modify the source address, Cisco provides a simple command to force the use of the interface address in all calls sent; this requirement is called *replacing the calling address*.

To suppress or replace the calling address, perform the appropriate task in interface configuration mode:

Task	Command
Suppress the calling (source) X.121 address in outgoing calls.	x25 suppress-calling-address
Replace the calling (source) X.121 address in switched calls.	x25 use-source-address

Suppress the Called Address

Some attachments require that no called (destination) address be presented in outgoing calls; this requirement is called *suppressing the called address*.

To suppress the called address, perform the following task in interface configuration mode:

Task	Command
Suppress the called (destination) X.121 address in outgoing calls.	x25 suppress-called-address

Establish a Default Virtual Circuit Protocol

The Call Request packet that sets up a virtual circuit can encode a field called the *Call User Data (CUD)* field. Typically the first few bytes of the CUD field identify which high-level protocol is carried by the virtual circuit. The router, when acting as a destination host, normally refuses a call if the CUD is absent or the protocol identification isn't recognized. The PAD protocol, however, specifies that unidentified calls be treated as PAD connection requests. Other applications require that they be treated as IP encapsulation connection requests, per RFC 877.

To configure either PAD or IP encapsulation treatment of unidentified calls, perform the following task in interface configuration mode:

Task	Command
Establish a default virtual circuit protocol.	<code>x25 default {ip pad}</code>

Disable Packet-Level Protocol (PLP) Restarts

By default, a Packet-Level Protocol (PLP) restart is performed when the link level resets (for example, when LAPB reconnects). Although PLP restarts can be disabled for those few networks that do not allow restarts, Cisco does not recommend disabling these restarts because doing so can cause anomalous packet layer behavior.



Caution Very few networks require this feature; Cisco does not recommend it be enabled except when attaching to a network that requires it.

To disable PLP restarts, perform the following task in interface configuration mode:

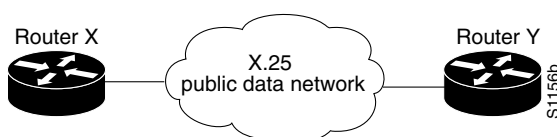
Task	Command
Disable packet-level restarts.	<code>no x25 linkrestart</code>

Configure an X.25 Datagram Transport

X.25 support is most commonly configured as a transport for datagrams across an X.25 network. Datagram transport (or *encapsulation*) is a cooperative effort between two hosts communicating across an X.25 network. You configure datagram transport by establishing a mapping on the encapsulating interface between the far host's protocol address (for example, IP or DECnet) and its X.121 address. Because the call identifies the protocol that the virtual circuit will carry (by encoding a Protocol Identifier, or PID, in the first few bytes of the CUD field), the terminating host can accept the call if it is configured to exchange the identified traffic with the source host.

Figure 25 illustrates two routers sending datagrams across an X.25 public data network (PDN).

Figure 25 Transporting LAN Protocols across an X.25 Public Data Network (PDN)



Perform the tasks in the following sections, as necessary, to complete the X.25 configuration for your network needs:

- Configure Subinterfaces
- Map Protocol Addresses to X.121 Addresses
- Establish an Encapsulation PVC
- Set X.25 TCP/IP Header Compression
- Configure X.25 Bridging

Configuring the X.25 parameters and special features, including payload compression and X.25 user facilities, are described in the section “Configure Additional X.25 Datagram Transport Features” later in this chapter.

Configure Subinterfaces

Subinterfaces are virtual interfaces that can be used to connect several networks to each other through a single physical interface. Subinterfaces are made available on Cisco routers because routing protocols, especially those using the split horizon principle, may need help to determine which hosts need a routing update. The split horizon principle, which allows routing updates to be distributed to other routed interfaces except the interface on which the routing update was received, works well in a LAN environment in which other routers reached by the interface have already received the routing update.

However, in a WAN environment using connection-oriented interfaces (like X.25 and Frame Relay), other routers reached by the same physical interface might not have received the routing update. Rather than forcing you to connect routers by separate physical interfaces, Cisco provides subinterfaces that are treated as separate interfaces. You can separate hosts into subinterfaces on a physical interface, the X.25 protocol is unaffected, and routing processes recognize each subinterface as a separate source of routing updates, so all subinterfaces are eligible to receive routing updates.

Understand Point-to-Point and Multipoint Subinterfaces

There are two types of subinterfaces: point-to-point and multipoint. Subinterfaces are implicitly multipoint unless configured as point-to-point.

A point-to-point subinterface is used to encapsulate one or more protocols between two hosts. An X.25 point-to-point subinterface will accept only a single encapsulation command (such as the **x25 map** or **x25 pvc** commands) for a given protocol, so there can be only one destination for the protocol. (However, you can use multiple encapsulation commands, one for each protocol, or multiple protocols for one map or PVC.) All protocol traffic routed to a point-to-point subinterface is forwarded to the one destination host defined for the protocol. (Because only one destination is defined for the interface, the routing process need not consult the destination address in the datagrams.)

A multipoint subinterface is used to connect one or more hosts for a given protocol. There is no restriction on the number of encapsulation commands that can be configured on a multipoint subinterface. Because the hosts appear on the same subinterface, they are not relying on the router to distribute routing updates between them. When a routing process forwards a datagram to a multipoint subinterface, the X.25 encapsulation process must be able to map the datagram’s destination address to a configured encapsulation command. If the routing process cannot find a map for the datagram destination address, the encapsulation will fail.

Note Because of the complex operations dependent on a subinterface and its type, the router will not allow a subinterface's type to be changed, nor can a subinterface with the same number be re-established once it has been deleted. After a subinterface has been deleted, you must reload the Cisco IOS software (by using the **reload** command) to remove all internal references. However, you can easily reconstitute the deleted subinterface by using a different subinterface number.

Create and Configure X.25 Subinterfaces

To create and configure a subinterface, complete the first task and one or both of the second tasks beginning in global configuration mode:

Task	Command
Create a point-to-point or multipoint subinterface.	interface serial <i>number.subinterface-number</i> [point-to-point multipoint]
Configure an X.25 encapsulation map for the subinterface. and/or Establish an encapsulation PVC for the subinterface.	x25 map <i>protocol address [protocol2 address2 [... [protocol9 address9]]] x121-address [option]</i> x25 pvc <i>circuit protocol address [protocol2 address2 [...[protocol9 address9]]] x121-address [option]</i>

For an example of configuring an X.25 subinterface and using multiple encapsulation commands for a single destination address, see the “Point-to-Point Subinterface Configuration Example” section later in this chapter.

For more general information about configuring subinterfaces, refer to the “Configuring Serial Interfaces” chapter in the *Configuration Fundamentals Configuration Guide*.

Note When configuring IP routing over X.25, you might need to make adjustments to accommodate split horizon effects. Refer to the “Configuring RIP” chapter in the *Network Protocols Configuration Guide, Part 1* for details about how the Cisco IOS software handles possible split horizon conflicts. By default, split horizon is enabled for X.25 attachments.

Map Protocol Addresses to X.121 Addresses

This section describes the X.25 single-protocol and multiprotocol encapsulation options that are available and describes how to map protocol addresses to an X.121 address for a remote host. This section also includes reference information about how protocols are identified.

Understand Protocol Encapsulation for Single-Protocol and Multiprotocol Virtual Circuits

Cisco has long supported encapsulation of a number of datagram protocols across X.25, using a standard method when available, or a proprietary method when necessary. These traditional methods assign a protocol to each virtual circuit. If more than one protocol is carried between the router and a given host, each active protocol will have at least one virtual circuit dedicated to carrying its datagrams.

Cisco also supports a newer standard, RFC 1356, which standardizes a method for encapsulating most datagram protocols over X.25. It also specifies how one virtual circuit can carry datagrams from more than one protocol.

The Cisco IOS software can be configured to use any of the available encapsulation methods with a particular host.

After you establish an encapsulation virtual circuit using any method, the Cisco IOS software sends and receives a datagram by simply fragmenting it into and reassembling it from an X.25 complete packet sequence. An X.25 complete packet sequence is one or more X.25 data packets that have the M-bit set in all but the last packet. A virtual circuit that can carry multiple protocols includes protocol identification data as well as the protocol data at the start of each complete packet sequence.

Understand Protocol Identification

This section contains background material only.

The various methods and protocols used in X.25 SVC encapsulation are identified in a specific field of the call packet; this field is defined by X.25 to carry Call User Data (CUD). Only PVCs do not use CUD to identify their encapsulation (since PVCs do not use the X.25 call setup procedures).

The primary difference between the available Cisco and IETF encapsulation methods is the specific value used to identify a protocol. When any of the methods establishes a virtual circuit for carrying a single protocol, the protocol is identified in the call packet by the CUD.

Table 3 summarizes the values used in the CUD field to identify protocols.

Table 3 Protocol Identification in the Call User Data (CUD) Field

Protocol	Cisco Protocol Identifier	IETF RFC 1356 Protocol Identifier
Apollo Domain	0xD4	0x80 (5-byte SNAP encoding ¹)
AppleTalk	0xD2	0x80 (5-byte SNAP encoding)
Banyan VINES	0xC0 00 80 C4 ²	0x80 (5-byte SNAP encoding)
Bridging	0xD5	(Not implemented)
ISO CLNS	0x81	0x81 ³
Compressed TCP	0xD8	0x00 (Multiprotocol) ⁴
DECnet	0xD0	0x80 (5-byte SNAP encoding)
IP	0xCC	0xCC ⁵ or 0x80 (5-byte SNAP encoding)
Novell IPX	0xD3	0x80 (5-byte SNAP encoding)
PAD	0x01 00 00 00 ⁶	0x01 00 00 00 ⁶
QLLC	0xC3	(Not available)
XNS	0xD1	0x80 (5-byte SNAP encoding)
Multiprotocol	(Not available)	0x00

1. Subnetwork Access Protocol (SNAP) encoding is defined from the Assigned Numbers RFC; Cisco's implementation recognizes only the IETF organizational unique identifier (OUI) 0x00 00 00 followed by a 2-byte Ethernet protocol type.

2. The use of 0xC0 00 80 C4 for Banyan VINES is defined by Banyan.

3. The use of 0x81 for CLNS is compatible with ISO/IEC 8473-3:1994.

4. Compressed TCP traffic has two types of datagrams, so IETF encapsulation requires a multiprotocol virtual circuit.

5. The use of 0xCC for IP is backwards-compatible with RFC 877.

6. The use of 0x01 00 00 00 for PAD is defined by ITU-T Recommendation X.29.

Once a multiprotocol virtual circuit has been established, datagrams on the virtual circuit have protocol identification data before the actual protocol data; the protocol identification values are the same used by RFC 1356 in the CUD field for an individual protocol.

Note IP datagrams can be identified with a 1-byte identification (0xCC) or a 6-byte identification (0x80 followed by the 5-byte SNAP encoding). The 1-byte encoding is used by default, although the SNAP encoding can be configured.

Map Datagram Addresses to X.25 Hosts

Encapsulation is a cooperative process between the router and another X.25 host. Because X.25 hosts are reached with an X.121 address (an X.121 address has between 0 and 15 decimal digits), the router must have a means to map a host's protocols and addresses to its X.121 address.

Each encapsulating X.25 interface must be configured with the relevant datagram parameters. For example, an interface that encapsulates IP will typically have an IP address.

A router set up for DDN or BFE service uses a dynamic mapping technique to convert between IP and X.121 addresses. These techniques have been designed specifically for attachment to the DDN network and to Blacker encryption equipment. Their design, restrictions, and operation make them work well for these specific applications, but not for other networks.

You must also establish the X.121 address of an encapsulating X.25 interface using the **x25 address** interface configuration command. This X.121 address is the address that encapsulation calls are directed to. This is also the source X.121 address used for originating an encapsulation call and is used by the destination host to map the source host and protocol to the protocol address. An encapsulation virtual circuit must be mapped at both the source and destination host interfaces. A DDN or BFE interface will have an X.121 address generated from the interface IP address, which for proper operation, should not be modified.

For each X.25 interface, you must explicitly map each destination host's protocols and addresses to its X.121 address. If needed and the destination host has the capability, one host map can be configured to support several protocols; alternatively, you can define one map for each supported protocol.

To establish a map, perform the following task in interface configuration mode:

Task	Command
Map one or more host protocol addresses to the host's X.121 address.	x25 map <i>protocol address [protocol2 address2[...[protocol9 address9]]]</i> <i>x121-address [option]</i>

For example, if you are encapsulating IP over a given X.25 interface, you must define an IP address for the interface and, for each of the desired destination hosts, map the host's IP address to its X.121 address.

Note You can map an X.121 address to as many as nine protocol addresses, but each protocol can be mapped only once in the command line.

An individual host map can use these keywords to specify the following protocols:

- **apollo**—Apollo Domain
- **appletalk**—AppleTalk
- **bridge**—Bridging
- **clns**—OSI Connectionless Network Service
- **compressedtcp**—TCP/IP header compression
- **decnet**—DECnet
- **ip**—IP
- **ipx**—Novell IPX
- **pad**—Packet Assembler/Disassembler
- **qllc**—IBM’s QLLC
- **vines**—Banyan VINES
- **xns**—XNS

Each mapped protocol, except bridging and CLNS, takes a datagram address. All bridged datagrams are either broadcast to all bridging destinations or are sent to a specific destination’s X.121 address, and CLNS uses the mapped X.121 address as the SNPA, which is referenced by a **clns neighbor** command. The configured datagram protocol(s) and their relevant address are mapped to the destination host’s X.121 address. All protocols that are supported for RFC 1356 operation can be specified in a single map. (Bridging and QLLC are not supported for RFC 1356 encapsulation.) If IP and TCP/IP header compression are both specified, the same IP address must be given for both protocols.

When setting up the address map, you can include options such as enabling broadcasts, specifying the number of virtual circuits allowed, and defining various user facility settings.

Note Multiprotocol maps, especially those configured to carry broadcast traffic, can result in significantly larger traffic loads, requiring a larger hold queue, larger window sizes, or multiple virtual circuits.

For specific information about how to establish a protocol to run over X.25, refer to the appropriate protocol chapters in the *Network Protocols Configuration Guide, Part 1, Network Protocols Configuration Guide, Part 2*, and *Network Protocols Configuration Guide, Part 3*.

You can simplify the configuration for the Open Shortest Path First (OSPF) protocol by adding the optional **broadcast** keyword. See the **x25 map** command description in the “X.25 and LAPB Commands” chapter of the *Wide-Area Networking Command Reference* for more information.

Configure PAD Access

By default, packet assembler/disassembler (PAD) connection attempts are processed for session creation or protocol translation (subject to the configuration of those functions) from all hosts. To restrict PAD connections to only statically mapped X.25 hosts, perform the following tasks in interface configuration mode:

Task	Command
Restrict PAD access.	x25 pad-access
Configure a host for PAD access.	x25 map pad <i>x121-address</i> [<i>option</i>]

You can configure outgoing PAD access using the optional features of the **x25 map pad** command without restricting incoming PAD connections to the configured hosts.

Establish an Encapsulation PVC

Permanent virtual circuits (PVCs) are the X.25 equivalent of leased lines; they are never disconnected. You do not need to configure an address map before defining a PVC; an encapsulation PVC implicitly defines a map.

To establish a PVC, perform the following task in interface configuration mode:

Task	Command
Set an encapsulation PVC.	x25 pvc <i>circuit protocol address</i> [<i>protocol2 address2</i> [... <i>[protocol9 address9]</i>]] <i>x121-address</i> [<i>option</i>]

The **x25 pvc** command uses the same protocol keywords as the **x25 map** command. See the “Map Datagram Addresses to X.25 Hosts” section of this chapter for a list of protocol keywords. Encapsulation PVCs also use a subset of the options defined for the **x25 map** command.

The user may establish multiple, parallel PVCs that carry the same set of encapsulation traffic by specifying the identical mappings for each PVC. Additionally, the user can permit a mixture of SVCs and PVCs to carry the traffic set by using the **x25 map** command to specify an **nvc count** that exceeds the number of configured PVCs. The total number of VCs, of whatever type, can never exceed 8.

For an example of configuring a PVC, see the section “PVC Used to Exchange IP Traffic Example” later in this chapter.

Set X.25 TCP/IP Header Compression

Cisco supports RFC 1144 TCP/IP header compression (THC) on serial lines using HDLC and X.25 encapsulation. THC encapsulation is only slightly different from other encapsulation traffic, but these differences are worth noting. The implementation of compressed TCP over X.25 uses one virtual circuit to pass the compressed packets. Any IP traffic (including standard TCP) is separate from THC traffic; it is carried over separate IP encapsulation virtual circuits or identified separately in a multiprotocol virtual circuit.

Note If you specify both **ip** and **compressedtcp** in the same **x25 map compressedtcp** command, they must both specify the same IP address.

To set up a separate virtual circuit for X.25 TCP/IP header compression, perform the following task in interface configuration mode:

Task	Command
Allow a separate virtual circuit for compressed packets.	x25 map compressedtcp <i>ip-address</i> [<i>protocol2 address2</i> [... <i>[protocol9 address9]</i>]] <i>x121-address</i> [<i>option</i>]

Configure X.25 Bridging

Cisco's transparent bridging software supports bridging over X.25 virtual circuits. Bridging is not supported for RFC 1356 operation. Bridge maps must include the **broadcast** option for correct operation.

To enable the X.25 bridging capability, perform the following task in interface configuration mode:

Task	Command
Define bridging of X.25 frames.	x25 map bridge <i>x121-address</i> broadcast [<i>option</i>]

Configure Additional X.25 Datagram Transport Features

The Cisco IOS software allows you to configure additional X.25 datagram transport features, including various user facilities defined for X.25 call setup.

This section describes the X.25 datagram transport features you can configure by using the options in the **x25 map** or **x25 pvc** encapsulation commands (or by setting an interface default). The tasks you perform depend upon your needs, the structure of your network, and the requirements of the service provider.

To configure the optional parameters, user facilities, and special features, perform one or more of the tasks described in the following sections:

- Configure X.25 Payload Compression
- Configure the Encapsulation Virtual Circuit Idle Time
- Increase the Number of Virtual Circuits Allowed
- Configure the Ignore Destination Time
- Establish the Packet Acknowledgment Policy
- Configure X.25 User Facilities
- Define the Virtual Circuit Packet Hold Queue Size
- Restrict Map Usage

Configure X.25 Payload Compression

For increased efficiency on relatively slow networks, the Cisco IOS software supports X.25 payload compression of outgoing encapsulation traffic.

The following restrictions apply to X.25 payload compression:

- The compressed virtual circuit must connect two Cisco routers, because X.25 payload compression is not standardized.

The data packets conform to the X.25 protocol rules, so a compressed virtual circuit can be switched through standard X.25 equipment. However, only Cisco routers can compress and decompress the data.

- Only datagram traffic can be compressed, although all the encapsulation methods supported by Cisco routers are available (for example, an IETF multiprotocol virtual circuit can be compressed).

Switched virtual circuits (SVCs) cannot be translated between compressed and uncompressed data, nor can PAD data be compressed.

- X.25 payload compression must be applied carefully.

Each compressed virtual circuit requires significant memory resources (for a dictionary of learned data patterns) and computation resources (every data packet received is decompressed and every data packet sent is compressed). Excessive use of compression can cause unacceptable overall performance.

- X.25 compression must be explicitly configured for a map command.

A received call that specifies compression will be rejected if the corresponding host map does not specify the **compress** option. An incoming call that does not specify compression can, however, be accepted by a map that specifies compression.

To enable payload compression over X.25, perform the following task in interface configuration mode:

Task	Command
Enable payload compression over X.25.	x25 map <i>protocol address</i> [<i>protocol2 address2</i> [...[<i>protocol9 address9</i>]]] <i>x121-address</i> compress [<i>option</i>]

This command specifies that X.25 compression is to be used between the two hosts. Because each virtual circuit established for compressed traffic uses significant amounts of memory, compression should be used with careful consideration of its impact on the performance.

The **compress** option may be specified for an encapsulation PVC.

Configure the Encapsulation Virtual Circuit Idle Time

The Cisco IOS software can clear a datagram transport or PAD SVC after a set period of inactivity. Routed SVCs are not timed for inactivity.

To set the time, perform the following tasks in interface configuration mode:

Task	Command
Set an idle time for clearing encapsulation.	x25 idle <i>minutes</i>
Specify an idle time for clearing a map's SVCs.	x25 map <i>protocol address</i> [<i>protocol2 address2</i> [...[<i>protocol9 address9</i>]]] <i>x121-address</i> idle <i>minutes</i>

For an example of configuring the SVC idle timer, see the section “Typical X.25 Configuration Example” later in this chapter. See the section “Monitor and Maintain LAPB and X.25” later in this chapter for additional commands that clear virtual circuits.

Increase the Number of Virtual Circuits Allowed

For X.25 datagram transport, you can establish up to eight VCs to one host for each map.

To increase the number of virtual circuits allowed, perform one or both of the following tasks in interface configuration mode:

Task	Command
Specify the default maximum number of SVCs that can be open simultaneously to one host for each map.	x25 nvc count
Specify the maximum number of SVCs allowed for a map.	x25 map protocol address [protocol2 address2 [...[protocol9 address9]]] x121-address nvc count

For an example of increasing the number of virtual circuits allowed, see the sections “Typical X.25 Configuration Example” and “DDN X.25 Configuration Example” later in this chapter.

Configure the Ignore Destination Time

Upon receiving a Clear for an outstanding datagram transport Call Request, the X.25 encapsulation code immediately tries another Call Request if it has more traffic to send. This action can overrun some X.25 switches.

To define the number of minutes the Cisco IOS software will prevent calls from going to a previously failed destination, perform the following task in interface configuration mode (incoming calls will still be accepted and cancel the timer):

Task	Command
Configure the ignore destination time.	x25 hold-vc-timer minutes

Establish the Packet Acknowledgment Policy

You can instruct the Cisco IOS software to send an acknowledgment packet when it has received a threshold of data packets it has not acknowledged, instead of waiting until its input window is full. A value of 1 sends an acknowledgment for each data packet received if it cannot be acknowledged in an outgoing data packet. This approach improves line responsiveness at the expense of bandwidth. A value of 0 restores the default behavior of waiting until the input window is full.

To establish the acknowledgment threshold, perform the following task in interface configuration mode:

Task	Command
Establish the threshold at which to acknowledge data packets.	x25 threshold packet-count

The packet acknowledgment threshold also applies to encapsulation PVCs.

Configure X.25 User Facilities

The X.25 software provides commands to support X.25 user facilities—options specified by the creators of the X.25 Recommendation—that allow you to use network features such as reverse charging, user identification, and flow control negotiation. You can choose to configure facilities on a per-map basis or on a per-interface basis. In the following table, the **x25 map** commands configure facilities on a per-map basis; the **x25 facility** commands specify the values sent for all encapsulation calls originated by the interface. Routed calls are not affected by the facilities specified for the outgoing interface.

To set the supported X.25 user facilities, perform one or more of the following tasks in interface configuration mode:

Task	Command
Select the closed user group.	x25 facility cug <i>group-number</i> or x25 map <i>protocol address [protocol2 address2 [...[protocol9 address9]]] x121-address cug group-number</i>
Set flow control parameter negotiation values to request on outgoing calls.	x25 facility packetsize <i>in-size out-size</i> or x25 map <i>protocol address [protocol2 address2 [...[protocol9 address9]]] x121-address packetsize in-size out-size</i> x25 facility windowsize <i>in-size out-size</i> or x25 map <i>protocol address [protocol2 address2 [...[protocol9 address9]]] x121-address windowsize in-size out-size</i>
Set reverse charging.	x25 facility reverse or x25 map <i>protocol address [protocol2 address2 [...[protocol9 address9]]] x121-address reverse</i>
Allow reverse charging acceptance.	x25 accept-reverse or x25 map <i>protocol address [protocol2 address2 [...[protocol9 address9]]] x121-address accept-reverse</i>
Select throughput class negotiation.	x25 facility throughput <i>in out</i> or x25 map <i>protocol address [protocol2 address2 [...[protocol9 address9]]] x121-address throughput in out</i>
Select transit delay.	x25 facility transit-delay <i>milliseconds</i> or x25 map <i>protocol address [protocol2 address2 [...[protocol9 address9]]] x121-address transit-delay milliseconds</i>
Set the Recognized Operating Agency (ROA) to use.	x25 facility roa <i>name</i> or x25 map <i>protocol address [protocol2 address2 [...[protocol9 address9]]] x121-address roa name</i>
Set the Cisco standard network user identification.	x25 map <i>protocol address [protocol2 address2 [...[protocol9 address9]]] x121-address nuid username password</i>
Set a user-defined network user identification allowing the format to be determined by your network administrator.	x25 map <i>protocol address [protocol2 address2 [...[protocol9 address9]]] x121-address nudata string</i>

The **window size** and **packet size** options are supported for PVCs, although they have a slightly different meaning because PVCs do not use the call setup procedure. If the PVC does not use the interface defaults for the flow control parameters, these options must be used to specify the values. Not all networks will allow a PVC to be defined with arbitrary flow control values.

Additionally, the D-bit is supported, if negotiated. PVCs allow the D-bit procedure because there is no call setup to negotiate its use. Both restricted and unrestricted fast select are also supported and are transparently handled by the software. No configuration is required for use of the D-bit or fast select facilities.

Define the Virtual Circuit Packet Hold Queue Size

To define the maximum number of packets that can be held while a virtual circuit is unable to send data, perform the following task in interface configuration mode:

Task	Command
Define the virtual circuit packet hold queue size.	x25 hold-queue <i>queue-size</i>

An encapsulation virtual circuit's hold queue size is determined when it is created; the **x25 hold-queue** command does not affect existing virtual circuits. This command also defines the hold queue size of encapsulation PVCs.

Restrict Map Usage

An X.25 map can be restricted so that it will not be used to place calls or so that it will not be considered when incoming calls are mapped.

To restrict X.25 map usage, use the following map options as needed in interface configuration mode:

Task	Command
Restrict incoming calls from a map.	x25 map <i>protocol address [protocol2 address2 [...[protocol9 address9]]] x121-address no-incoming</i>
Restrict outgoing calls from a map.	x25 map <i>protocol address [protocol2 address2 [...[protocol9 address9]]] x121-address no-outgoing</i>

Configure Priority Queuing or Custom Queuing for X.25

Two types of output queuing are available for X.25:

- Priority queuing—Classifies packets based on certain criteria and then assigns the packets to one of four output queues, with high, medium, normal, or low priority.
- Custom queuing—Classifies packets, assigns them to one of 16 output queues, and controls the percentage of an interface's available bandwidth that is used for a queue.

Understand the Effect of X.25 Flow Control on Queuing

Output queuing for X.25 interfaces differs subtly from its use with other protocols because X.25 is a strongly flow-controlled protocol. Each X.25 virtual circuit has an authorized number of packets it can send before it must suspend transmission to await acknowledgment of one or more of the packets that were sent.

Queue processing is also subject to a virtual circuit's ability to send data; a high priority packet on a virtual circuit that cannot send data will not stop other packets from being sent if they are queued for a virtual circuit that can send data. In addition, a datagram that is in the process of being fragmented and sent may have its priority artificially promoted if higher priority traffic is blocked by the fragmentation operation.

Configure Queuing

Both priority queuing and custom queuing can be defined, but only one method can be active on a given interface.

To configure priority queuing and custom queuing for X.25, perform the following tasks:

- Step 1** Perform the standard priority and custom queuing tasks *except* the task of assigning a priority or custom group to the interface, as described in the “Managing System Performance” chapter in the *Configuration Fundamentals Configuration Guide*.
- Step 2** Perform the standard X.25 encapsulation tasks, as specified in the “Configure an X.25 Datagram Transport” section earlier in this chapter.
- Step 3** Assign either a priority group or a custom queue to the interface, as described in the “Managing System Performance” chapter in the *Configuration Fundamentals Configuration Guide*.

Note Connection-oriented virtual circuits (for example, QLLC, PAD, switched X.25) will use the interface's default queue. To maintain the correct order, all connection-oriented VCs use a single output queue for sending data.

Configure X.25 Routing

The X.25 software implementation allows virtual circuits to be routed from one X.25 interface to another and from one router to another. The routing behavior can be controlled with switching and X.25-over-TCP (XOT) configuration commands, based on a locally built table.

X.25 encapsulation can share an X.25 serial interface with the X.25 switching support. Switching or forwarding of X.25 virtual circuits can be done two ways:

- Incoming calls received from a local serial interface running X.25 can be forwarded to another local serial interface running X.25. This is known as *local X.25 switching* because the router handles the complete path. It does not matter whether the interfaces are configured as DTE or DCE devices, because the software takes the appropriate actions.
- An incoming call can also be forwarded using the *XOT* service (previously *remote switching* or *tunneling*). Upon receipt of an incoming X.25 call, a TCP connection is established to the destination XOT host (for example, another Cisco router) that will, in turn, handle the call using its own criteria. All X.25 packets are sent and received over the reliable TCP data stream. Flow control is maintained end-to-end. It does not matter whether the interface is configured for DTE or DCE, because the software takes the appropriate actions.

Running X.25 over TCP/IP provides a number of benefits. The datagram containing the X.25 packet can be switched by other routers using their high-speed switching abilities. X.25 connections can be sent over networks running only the TCP/IP protocols. The TCP/IP protocol suite runs over many different networking technologies, including Ethernet, Token Ring, T1 serial, and FDDI. Thus X.25 data can be forwarded over these media to another router, where it can, for example, be switched to an X.25 interface.

When the connection is made locally, the switching configuration is used; when the connection is across a LAN, the XOT configuration is used. The basic function is the same for both types of connections, but different configuration commands are required for each type of connection.

The X.25 switching subsystem supports the following facilities and parameters:

- D-bit negotiation (data packets with the D-bit set are passed through transparently)
- Variable-length interrupt data (if not operating as a DDN or BFE interface)
- Flow control parameter negotiation:
 - Window size up to 7, or 127 for modulo 128 operation
 - Packet size up to 4096 (if the LAPB layers used are capable of handling the requested size)
- Basic closed user group selection
- Throughput class negotiation
- Reverse charging and fast select

The handing of these facilities is described in the “X.25 Facility Handling” appendix in the *Wide-Area Networking Command Reference*.

To configure X.25 routing, perform the tasks in the following sections:

- Enable X.25 Routing
- Configure an X.25 Route
- Configure a PVC Switched between X.25 Interfaces
- Configure X.25 Switching between PVCs and SVCs

You may also need to configure additional X.25 routing features, as required for your network. Each task is described in a following section.

Enable X.25 Routing

You must enable X.25 routing to use switch VCs.

To enable X.25 routing, perform the following task in global configuration mode:

Task	Command
Enable X.25 routing.	x25 routing [use-tcp-if-defs]

The **use-tcp-if-defs** keyword is used by some routers that receive remote routed calls from older versions of XOT; it might be needed if the originating router cannot be migrated to a new software release. The use of this keyword is described in the “Configure XOT to Use Interface Default Flow Control Values” section later in this chapter.

For an example of configuring X.25 routing, see the sections “X.25 Route Address Pattern Matching Example” and “X.25 Routing Examples” later in this chapter.

Configure an X.25 Route

An X.25 route table enables you to control which destination is selected for several applications. When an X.25 service receives a call that must be forwarded, the X.25 route table determines which X.25 service (X.25, CMNS or XOT) and destination should be used. When a PAD call is originated by the router, either from a user request or a protocol translation event, the route table similarly determines what X.25 service and destination should be used.

You create the X.25 route table and add route entries to it. You can optionally specify the entry's order in the table, the criteria to match against the virtual circuit information, and whether to modify the destination or source addresses. Each entry must specify the disposition of the virtual circuit (that is, what is done with the virtual circuit). Each route can also specify XOT keepalive options.

The route table is used as follows:

- Virtual circuit information is matched against selection criteria specified for each route.
- The table is scanned sequentially from the top.
- The first matching route determines how the virtual circuit is handled.
- Once a matching entry is found, the call addresses can be modified and the call is disposed of (forwarded or cleared) as instructed by the entry.

Each application can define special conditions if a route will not be used or what occurs if no route matches. For instance, switched X.25 will skip a route if the disposition interface is down and clear a call if no route matches.

To configure an X.25 route (thus adding the route to the X.25 routing table), perform the following task in global configuration mode:

Task	Command
Configure an X.25 route.	x25 route [# <i>position</i>] {[<i>selection</i>] [<i>modification</i>]} <i>disposition</i> [<i>xot-keepalive</i>]

The following options offer versatility and flexibility when you use the **x25 route** command:

- *#position*—Position in the table. You can use the optional *#position* element to indicate the number of the entry in the route table. For example, #9 indicates the ninth entry from the top. The route table is always searched sequentially from the top, and the first match found will be used.
- *selection*—Criteria to define which virtual circuits the route will apply to. You can match against zero to four of the following optional *selection* elements:
 - *destination-pattern*
 - **source** *source-pattern*
 - **dest-ext** *nsap-destination-pattern*
 - **culd** *user-data-pattern*
- *modification*—Modifications to the source or destination address for address translation. You can use none, one, or both of the following optional *modification* elements to change the source or destination address before forwarding the call to the destination:
 - **substitute-source** *rewrite-source*
 - **substitute-dest** *rewrite-destination*

Note You must include a selection option or a modification option in an **x25 route** command.

- *disposition*—Where the virtual circuit will be forwarded or whether it will be cleared. You are required to use one of the following *disposition* elements:
 - **interface** *serial-interface*

A route to a specific *serial-interface* will send the virtual circuit to an X.25 service on a synchronous serial interface.
 - **interface** *cmns-interface* **mac** *mac-address*

A route to a broadcast interface will send the virtual circuit to a CMNS partner reachable on a broadcast medium at a specified MAC address. The CMNS interface can be an Ethernet, Token Ring, or FDDI interface.
 - **xot** *ip-address* [*ip2-address* [...*ip6-address*]] [**xot-source** *interface*]

A route to an **xot** destination (formerly called a remote or tunneled configuration) will send the virtual circuit to the XOT service for establishment of a TCP connection across which the XOT virtual circuit packets will travel. An **xot** disposition may specify alternate destinations to try if a TCP connection cannot be established for all preceding destinations.
 - **clear**

A route to a **clear** destination will deny further service to the virtual circuit by shutting down the connection.
- *xot-keepalive*—Options. You can use none, one, or both of the following optional *xot-keepalive* elements:
 - **xot-keepalive-period** *seconds*
 - **xot-keepalive-tries** *count*

Configure a PVC Switched between X.25 Interfaces

You can configure an X.25 PVC in the X.25 switching software. As a result, DTEs that require permanent circuits can be connected to a router acting as an X.25 switch and have a properly functioning connection. X.25 resets will be sent to indicate when the circuit comes up or goes down. Both interfaces must define complementary locally switched PVCs.

To configure a locally switched PVC, perform the following task in interface configuration mode:

Task	Command
Configure a locally switched PVC.	x25 pvc <i>number1</i> interface <i>type</i> <i>number</i> pvc <i>number2</i> [<i>option</i>]

The command options are **packetsize** *in out* and **window** *size in out*; they allow a PVC's flow control values to be defined if they differ from the interface defaults.

For an example of configuring a locally switched PVC, see the section “PVC Switching on the Same Router Example” later in this chapter.

To ensure that these TCP sessions remain connected in the absence of XOT traffic, perform the following task in global configuration mode:

Task	Command
Enable keepalives for TCP sessions sent and received to ensure timely detection of a connection failure.	service tcp-keepalives-in service tcp-keepalives-out

TCP keepalives also inform a router when an XOT SVCs session is not active, thus freeing router resources.

For examples of enabling keepalives, see the “Simple Switching of a PVC over XOT Example” and “PVC Switching over XOT Example” sections later in this chapter.

Configure X.25 Switching between PVCs and SVCs

To configure PVC to SVC switching between two serial interfaces, both interfaces must already be configured for X.25. In addition, X.25 switching must be enabled using the **x25 routing** global configuration command. The PVC interface must be a serial interface configured with X.25 encapsulation. (The SVC interface may use X.25, XOT, or CMNS.)

Once the interfaces have been configured for X.25 switching, perform the following task in interface configuration mode:

Task	Command
Configure the PVC whose traffic is to be forwarded to an SVC.	x25 pvc number1 svc x121-address <i>[flow-control-options] [call-control-options]</i>

To display information about the switched PVC to SVC circuit, perform the following task in EXEC mode:

Task	Command
Display information about the active SVCs and PVCs.	show x25 vc [lcn]

For an example of configuring switching between a PVC and SVC, see the section “X.25 Switching between PVCs and SVCs Example” later in this chapter.

Configure Additional X.25 Routing Features

To configure other, less common X.25 routing features, perform the tasks in the following sections:

- Configure XOT to Use Interface Default Flow Control Values
- Substitute Addresses in an X.25 Route
- Configure XOT Alternate Destinations

Configure XOT to Use Interface Default Flow Control Values

When setting up a connection, the source and destination XOT implementations need to cooperate to determine the flow control values that apply to the SVC. The source XOT ensures cooperation by encoding the X.25 flow control facilities (the window sizes and maximum packet sizes) in the X.25 Call packet; the far host’s XOT implementation can then correctly negotiate the flow control values at the destination interface and, if needed, indicate the final values in the X.25 Call Confirm packet.

When XOT receives a call that leaves one or both flow control values unspecified, it supplies the values. The values supplied are a window size of 2 packets and maximum packet size of 128 bytes; according to the standards, any SVC can be negotiated to use these values. Thus when XOT receives a call from an older XOT implementation, it can specify in the Call Confirm packet that these flow control values must revert to the lowest common denominator.

What the older XOT implementations required was that the source and destination XOT router use the same default flow control values on the two X.25 interfaces that connect the SVC. Consequently, connections with mismatched flow control values were created when this assumption was not true, which resulted in mysterious problems. The current implementation's practice of signaling the values in the Call Confirm packet avoids these problems.

Occasionally the older XOT implementation will be connected to a piece of X.25 equipment that cannot handle modification of the flow control parameters in the Call Confirm packet. These configurations should be upgraded to use a more recent version of XOT; when upgrade is not possible, XOT's behavior causes a migration problem. In this situation, you may configure the Cisco IOS software to cause XOT to obtain unspecified flow control facility values from the destination interface's default values.

To configure this behavior, add the option **use-tcp-if-defs** when enabling X.25 routing in global configuration mode:

Task	Command
Enable X.25 routing; optionally modify XOT's source of unencoded flow control values.	x25 routing [use-tcp-if-defs]

Substitute Addresses in an X.25 Route

When interconnecting two separate X.25 networks, you must sometimes provide for address substitution for routes. The **x25 route** command supports modification of X.25 source and destination addresses.

To modify addresses, perform either or both of the following tasks in global configuration mode:

Task	Command
Modify the X.25 source address.	x25 route [#position] destination-pattern [source source-pattern] [substitute-source rewrite-source] interface interface number
Modify the X.25 destination address.	x25 route [#position] destination-pattern [source source-pattern] [substitute-dest rewrite-dest] interface interface number

Address substitution is available for all applications of X.25 routes.

Configure XOT Alternate Destinations

Routes to XOT hosts can be configured with alternate destination hosts. On routing a call, XOT will try each XOT destination host in sequence; if the TCP connection attempt fails, the next destination will be tried. Up to six XOT destination addresses can be entered.

To configure an XOT route with alternate destinations, (thus adding it to the X.25 routing table), perform the following task in global configuration mode:

Task	Command
Configure an XOT route; optionally define alternate XOT destination hosts.	<code>x25 route [#position] destination-pattern xot ip-address [ip-address2... [ip-address6]]</code>

The sequence of alternate destination XOT host addresses is simply added to the normal XOT route configuration command.

Note It can take up to 50 seconds to try an alternate route due to TCP timings.

For an example of constructing the routing table, see the section “X.25 Routing Examples” later in this chapter.

Configure CMNS Routing

The Connection-Mode Network Service (CMNS) provides a mechanism through which X.25 services can be extended to nonserial media through the use of packet-level X.25 over frame-level LLC2.

Note For information about configuring LLC2 parameters, refer to the “Configuring SDLC and LLC2 Parameters” chapter in the *Bridging and IBM Networking Configuration Guide*.

Cisco’s CMNS implementation permits most X.25 services to be extended across a LAN, although datagram encapsulation and QLLC operations are not available. For example, a DTE host and a Sun workstation can be interconnected via the router’s LAN interfaces *and* to a remote OSI-based DTE through a WAN interface to an X.25 packet-switched network (PSN).

Implementing CMNS routing involves completing the tasks in the following sections:

- Enable CMNS on an Interface
- Configure a Route to a CMNS Host

Enable CMNS on an Interface

To enable CMNS on a nonserial interface, perform the following task in interface configuration mode:

Task	Command
Enable CMNS.	<code>cmns enable</code>

For an example of enabling CMNS on an interface, see the section “CMNS Switching Example” later in this chapter.

Configure a Route to a CMNS Host

Once CMNS is enabled on a nonserial interface, the router can forward calls over that medium by configuring **x25 route** commands that define the MAC address of each CMNS host that can be reached.

To define routes to CMNS hosts, perform the following task in interface configuration mode:

Task	Command
Define a route to a CMNS host.	x25 route <i>match-criteria</i> interface <i>cmns-interface</i> mac <i>mac-address</i>

Configure DDN or BFE X.25

The Defense Data Network (DDN) X.25 protocol has two versions: Basic Service and Standard Service. Cisco's X.25 implementation supports only the Standard Service and also includes Blacker Front End (BFE) and Blacker Emergency Mode operation.

DDN X.25 Standard Service requires that the X.25 data packets carry IP datagrams. The DDN packet switching nodes (PSNs) can extract the IP datagram from within the X.25 packet and pass data to another Standard Service host.

The DDN X.25 Standard is the required protocol for use with DDN PSNs. The Defense Communications Agency (DCA) has certified Cisco's DDN X.25 Standard implementation for attachment to the Defense Data Network. As part of the certification, Cisco IOS software is required to provide a scheme for dynamically mapping Internet addresses to X.121 addresses. See the section "Understand DDN X.25 Dynamic Mapping" that follows for details on that scheme.

To enable DDN X.25 service, perform the tasks in the following sections:

- Enable DDN X.25
- Define IP Precedence Handling

To enable BFE X.25 service, perform the task in the following section:

- Configure Blacker Front End (BFE) X.25

Understand DDN X.25 Dynamic Mapping

The DDN X.25 standard implementation includes a scheme for dynamically mapping all classes of IP addresses to X.121 addresses without a table. This scheme requires that the IP and X.121 addresses conform to the formats shown in Figure 26 and Figure 27. These formats segment the IP addresses into network (N), host (H), logical address (L), and PSN (P) portions. For the BFE encapsulation, the IP address is segmented into Port (P), Domain (D), and BFE ID number (B). The DDN algorithm requires that the host value be less than 64.

Figure 26 DDN IP Address Conventions

Class A:	Net.Host.LH.PSN → 0000 0 PPPHH00
Bits:	8 8 8 8
Class B:	Net.Net.Host.PSN → 0000 0 PPPHH00
Bits:	8 8 8 8
Class C:	Net.Net.Net.Host.PSN → 0000 0 PPPHH00
Bits:	8 8 8 4 4

S2302

Figure 27 BFE IP Address Conventions

BFE Class A :	Net.unused.Port.Domain.BFE → 0000 0 PDDDBBB
Bits:	8 1 3 10 10

S2823

The DDN conversion scheme uses the host and PSN portions of an IP address to create the corresponding X.121 address. The DDN conversion mechanism is limited to Class A IP addresses; however, the Cisco IOS software can convert Class B and Class C addresses as well. As indicated, this method uses the last two octets of a Class B address as the host and PSN identifiers, and the upper and lower four bits in the last octet of a Class C address as the host and PSN identifiers, respectively. The BFE conversion scheme requires a Class A IP address.

The DDN conversion scheme uses a physical address mapping if the host identifier is numerically less than 64. (This limit derives from the fact that a PSN cannot support more than 64 nodes.) If the host identifier is numerically larger than 64, the resulting X.121 address is called a *logical address*. The DDN does not use logical addresses.

The format of physical DDN X.25/X.121 addresses is ZZZZFIIHHZZ(SS). Each character represents a digit and is described in the following list:

- ZZZZ represents four zeros.
- F is zero to indicate a physical address.
- III represents the PSN octet from the IP address padded with leading zeros.
- HH is the host octet from the IP address padded with leading zeros.
- ZZ represents two zeros.
- (SS) represents the optional and unused subaddress.

The physical and logical mappings of the DDN conversion scheme always generate a 12-digit X.121 address. Subaddresses are optional; when added to this scheme, the result is a 14-digit X.121 address. The DDN does not use subaddressing.

Packets using routing and other protocols that require broadcast support can successfully traverse X.25 networks, including the DDN. This traversal requires the use of network protocol-to-X.121 maps, because the router must know explicitly where to deliver broadcast datagrams. (X.25 does not support broadcasts.) You can mark network protocol-to-X.121 map entries to accept broadcast packets; the router then sends broadcast packets to hosts with marked entries. For DDN or BFE operation, the router generates the interface X.121 addresses from the interface IP address using the DDN or BFE mapping technique.

Enable DDN X.25

Both DCE and DTE operation causes the Cisco IOS software to specify the Standard Service facility in the Call Request packet, which notifies the PSNs to use Standard Service.

To enable DDN X.25, perform one of the following tasks in interface configuration mode, as appropriate for your network:

Task	Command
Set DDN X.25 DTE operation.	encapsulation x25 ddn
Set DDN X.25 DCE operation.	encapsulation x25 dce ddn

For an example of enabling DDN X.25, see the section “DDN X.25 Configuration Example” later in this chapter.

Define IP Precedence Handling

Using Standard Service, the DDN can be configured to provide separate service for datagrams with high precedence values. When IP precedence handling is enabled, the router uses a separate X.25 SVC to handle each of four precedence classes of IP traffic—routine, priority, immediate, and other. An IP datagram is transmitted across an SVC that is carrying the appropriate precedence only.

By default, the DDN X.25 software opens one virtual circuit for all types of service values. You can enable the precedence-sensitivity feature by performing the following task in interface configuration mode:

Task	Command
Allow a new virtual circuit based on the type of service (TOS) field.	x25 ip-precedence

Verify that your host does not send nonstandard data in the TOS field. Nonstandard data can cause multiple, wasteful virtual circuits to be created.

Configure Blacker Front End (BFE) X.25

For environments that require a high level of security, the Cisco IOS software supports attachment to Defense Data Network (DDN) Blacker Front End (BFE) equipment and Blacker Emergency Mode operation.

Blacker Emergency Mode allows your BFE device and your router to function in emergency situations. When the router is configured to participate in emergency mode and the BFE device is in emergency mode, the Cisco IOS software sends address translation information to the BFE device to assist it in sending information.

Cisco’s implementation of Blacker Emergency Mode adheres to the specifications outlined in the DCA Blacker Interface Control document, published March 21, 1989.

Your BFE device can be configured to respond in one of the following ways:

- Enters emergency mode when requested to by the network. If the Cisco IOS software is configured to respond to a BFE device in emergency mode, or if the EXEC command **bfe enter** is used, the software sends address translation information to the BFE device.
- Never enters emergency mode.

- Notifies the router that the emergency mode window is open and waits for the router to signal it to enter emergency mode. If the software is configured to respond to a BFE in emergency mode, or if the EXEC command **bfe enter** is used, the software sends a special address translation packet to the BFE device. The “special” data includes a command to the BFE to enter emergency mode.

To configure Blacker Emergency Mode, complete the tasks in the following sections:

- Set BFE Encapsulation
- Provide Address Translation
- Define Emergency Conditions
- Enter Blacker Emergency Mode

For an example of configuring Blacker Emergency mode, see the section “Blacker Emergency Mode Example” at the end of this chapter.

Set BFE Encapsulation

BFE encapsulation operates to map between Class A IP addresses and the X.121 addresses expected by the BFE encryption device.

To set BFE encapsulation on the router attached to a BFE device, perform the following task in interface configuration mode:

Task	Command
Set BFE encapsulation on the router attached to a BFE device.	encapsulation x25 bfe

Provide Address Translation

You must set up a table that provides the address translation information the router sends to the BFE device when the BFE device is in emergency mode.

To provide address translation information to the BFE device, perform the following task in interface configuration mode:

Task	Command
Set up the table that lists the BFE nodes (host or gateways) to which the router will send packets.	x25 remote-red <i>host-ip-address</i> remote-black <i>blacker-ip-address</i>

Define Emergency Conditions

To define the circumstances under which the router participates in emergency mode and how it will participate, perform the following tasks in interface configuration mode:

Task	Command
Define the circumstances under which the router will participate in emergency mode.	x25 bfe-emergency { never always decision }
Define how a router configured as x25 bfe-emergency decision will participate in emergency mode.	x25 bfe-decision { no yes ask }

Enter Blacker Emergency Mode

To set the router to participate in emergency mode or to end participation in emergency mode when your router is so configured, perform the following task in EXEC mode:

Task	Command
Set router to participate in emergency mode.	bfe {enter leave} <i>type number</i>

Create X.29 Access Lists

Protocol translation software supports access lists, which make it possible to limit access to the access server from 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 following tasks:

Step 1 Create an X.29 access list.

Step 2 Apply an access list to a virtual terminal line or to protocol translation.

These tasks are described in the following sections.

When configuring protocol translation, you can specify an access list number with each **translate** command. When translation sessions result from incoming PAD connections, the corresponding X.29 access list is used. Refer to the chapter “Protocol Translation and Virtual Asynchronous Device Commands” in the *Dial Solutions Command Reference* for more information about the **translate** command.

For an example of defining an X.29 access list, see the section “X.29 Access List Example” at the end of this chapter.

Create an Access List

To specify the access conditions, perform the following global configuration task:

Task	Command
Restrict incoming and outgoing connections between a particular virtual terminal line (into a Cisco access server) and the addresses in an access list.	x29 access-list <i>access-list-number</i> {deny permit} <i>x121-address</i>

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 is denied.

Apply an Access List to a Line

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

Task	Command
Restrict incoming and outgoing connections between a particular virtual terminal line (into a Cisco access server) and the addresses in an access list.	access-class <i>access-list-number</i> in

The access list number is used for incoming TCP access, incoming local-area transport (LAT) access, and for incoming PAD access. For TCP access, the protocol translator uses the defined IP access lists. For LAT access, the protocol translator uses the defined LAT access list. For incoming PAD connections, an X.29 access list is used. If you want to have access restrictions only on one of the protocols, then you can create an access list that permits all addresses for the other protocol.

Create an X.29 Profile Script

You can create an X.29 profile script for use by the **translate** command. When an X.25 connection is established, the protocol translator then acts as if an X.29 Set Parameter packet had been sent that contained the parameters and values set by this command.

To create an X.29 profile script, perform the following global configuration task:

Task	Command
Create an X.29 profile script.	x29 profile { default <i>name</i> } <i>parameter:value</i> [<i>parameter:value</i>]

For an example of a profile script, see the section “X.29 Profile Script Example” at the end of this chapter.

Monitor and Maintain LAPB and X.25

To monitor and maintain X.25 and LAPB, perform any of the following tasks in EXEC mode:

Task	Command
Clear an SVC, restart an X.25 or CMNS service, or reset a PVC.	clear x25 { <i>serial number</i> <i>cmns-interface mac-address</i> } [<i>vc-number</i>]
Clear an XOT SVC or reset an XOT PVC.	clear xot remote <i>ip-address port</i> local <i>ip-address port</i>
Display CMNS information.	show cmns [<i>type number</i>]
Display operation statistics for an interface.	show interfaces serial <i>number</i>
Display CMNS connections over LLC2.	show llc2
Display information about VCs on an X.25 interface (a serial interface) or a CMNS interface (an Ethernet, Token Ring, or FDDI interface).	show x25 interface [<i>serial number</i> <i>cmns-interface mac mac-address</i>]
Display the protocol-to-X.121 address map.	show x25 map
Display the one-to-one mapping of the host IP addresses and the remote BFE device’s IP addresses.	show x25 remote-red
Display routes assigned by the x25 route command.	show x25 route
Display information about X.25 services.	show x25 services
Display details of active virtual circuits.	show x25 vc [<i>lcn</i>]
Display information for all XOT virtual circuits or, optionally, for the virtual circuits that match a specified set of criteria.	show x25 xot [local <i>ip-address</i> [port <i>port</i>]] [remote <i>ip-address</i> [port <i>port</i>]]

Note See the “X.25 Cause and Diagnostic Codes” appendix in the *Debug Command Reference* for a description of PVC states that can appear in these **show command** displays.

X.25 and LAPB Configuration Examples

The following sections provide examples to help you understand how to configure LAPB and X.25 for your network:

- Typical LAPB Configuration Example
- Transparent Bridging for Multiprotocol LAPB Encapsulation Example
- Typical X.25 Configuration Example
- Virtual Circuit Ranges Example
- PVC Switching on the Same Router Example
- X.25 Route Address Pattern Matching Example
- X.25 Routing Examples
- PVC Used to Exchange IP Traffic Example
- Point-to-Point Subinterface Configuration Example
- Simple Switching of a PVC over XOT Example
- PVC Switching over XOT Example
- X.25 Switching between PVCs and SVCs Example
- CMNS Switching Example
- CMNS Switched over a PDN Example
- CMNS Switched over Leased Lines Example
- DDN X.25 Configuration Example
- Blacker Emergency Mode Example
- X.25 Configured to Allow Ping Support over Multiple Lines Example
- Booting from a Network Server over X.25 Example
- X.29 Access List Example
- X.29 Profile Script Example

Typical LAPB Configuration Example

In the following example, the frame size (N1), window size (k), and maximum retransmission (N2) parameters retain their default values. The **encapsulation** interface configuration command sets DCE operation to carry a single protocol, IP by default. The **lapb t1** interface configuration command sets the retransmission timer to 4,000 milliseconds (4 seconds) for a link with a long delay or slow connecting DTE device.

```
interface serial 3
  encapsulation lapb dce
  lapb t1 4000
```

Transparent Bridging for Multiprotocol LAPB Encapsulation Example

The following example configures transparent bridging for multiprotocol LAPB encapsulation:

```
no ip routing
!
interface Ethernet 1
  no ip address
  no mop enabled
  bridge-group 1
!
interface serial 0
  no ip address
  encapsulation lapb multi
  bridge-group 1
!
bridge 1 protocol ieee
```

Typical X.25 Configuration Example

The following example shows the complete configuration for a serial interface connected to a commercial X.25 PDN for routing the IP protocol. The IP subnetwork address 172.25.9.0 has been assigned for the X.25 network.

Note When you are routing IP over X.25, you must treat the X.25 network as a single IP network or subnetwork. Map entries for routers with addresses on subnetworks other than the one on which the interface's IP address is stored are ignored by the routing software. Additionally, all routers using the subnet number must have map entries for all others routers. Moreover, using the broadcast option with dynamic routing can result in significantly larger traffic loads, requiring a larger hold queue, larger window sizes, or multiple virtual circuits.

```
interface serial 2
  ip address 172.25.9.1 255.255.255.0
!
  encapsulation X25
!
! The "bandwidth" command is not part of the X.25
! configuration; it is especially important to understand that it does not
! have any connection with the X.25 entity of the same name.
! "bandwidth" commands are used by IP routing processes (currently only IGRP)
! to determine which lines are the best choices for traffic.
! Since the default is 1544 Kbaud, and X.25 service at that rate is not generally
! available, most X.25 interfaces that are being used with IGRP in a
! real environment will have "bandwidth" settings.
!
! This is a 9.6 Kbaud line:
!
  bandwidth 10
! You must specify an X.121 address to be assigned to the X.25 interface by the PDN.
!
  x25 address 31370054065
!
! The following Level 3 parameters have been set to match the network.
! You generally need to change some Level 3 parameters, most often
! those listed below. You might not need to change any Level 2
! parameters, however.
!
```

```

x25 htc 32
!
! These Level 3 parameters are default flow control values; they need to
! match the PDN defaults. The values used by an SVC are negotiable on a per-call basis:
!
x25 win 7
x25 wout 7
x25 ips 512
x25 ops 512
!
!
! The following commands configure the default behavior for our encapsulation
! SVCs
!
x25 idle 5
x25 nvc 2
!
! The following commands configure the X.25 map. If you want to exchange
! routing updates with any of the routers, they would need
! "broadcast" flags.
! If the X.25 network is the only path to the routers, static routes are
! generally used to save on packet charges. If there is a redundant
! path, it might be desirable to run a dynamic routing protocol.
!
x25 map IP 172.25.9.3 31370019134 ACCEPT-REVERSE
! ACCEPT-REVERSE allows collect calls
x25 map IP 172.25.9.2 31370053087
!
! If the PDN cannot handle fast back-to-back frames, use the
!"transmitter-delay" command to slow down the interface.
!
transmitter-delay 1000

```

Virtual Circuit Ranges Example

The following example sets the virtual circuit ranges of 5 to 20 for incoming calls only (from the DCE to the DTE) and 25 to 1024 for either incoming or outgoing calls. It also specifies no virtual circuits are reserved for outgoing calls (from the DTE to the DCE). Up to four permanent virtual circuits can be defined on virtual circuits 1 through 4.

```

x25 lic 5
x25 hic 20
x25 ltc 25

```

PVC Switching on the Same Router Example

In the following example, a PVC is connected between two serial interfaces on the same router. In this type of interconnection configuration, the destination interface must be specified along with the PVC number on that interface. To make a working PVC connection, two commands must be specified, each pointing to the other.

```

interface serial 0
  encapsulation x25
  x25 ltc 5
  x25 pvc 1 interface serial 1 pvc 4
!
interface serial 1
  encapsulation x25
  x25 ltc 5
  x25 pvc 4 interface serial 0 pvc 1

```

X.25 Route Address Pattern Matching Example

The following example shows how to route X.25 calls with addresses whose first four Data Network Identification Code (DNIC) digits are 1111 to interface serial 3, and to change the DNIC field in the addresses presented to the equipment connected to that interface to 2222. The `\` in the rewrite pattern indicates the portion of the original address matched by the digits following the 1111 DNIC.

```
x25 route ^1111(.*) substitute-dest 2222\1 interface serial 3
```

Figure 28 shows a more contrived command intended to illustrate the power of the rewriting scheme.

Figure 28 X.25 Route Address Pattern Matching Example

```
x25 route ^(...)..(..)..(..)(..)$ substitute-dest \2\4\3\1 interface serial 0
```

The command in Figure 28 causes all X.25 calls with 14-digit called addresses to be routed through interface serial 0. The incoming DNIC field is moved to the end of the address. The fifth, sixth, ninth, and tenth digits are deleted, and the thirteenth and fourteenth are moved before the eleventh and twelfth.

X.25 Routing Examples

The following examples illustrate how to enable the X.25 switch service, and how to configure a router on a Tymnet/PAD switch to accept and forward calls.

This first example shows how to enable X.25 switching, as well as how to enter routes into the X.25 routing table:

```
! Enable X.25 forwarding
x25 routing
!
! Enter routes into the table. Without a positional parameter, entries
! are appended to the end of the table
x25 route ^100$ interface serial 0
x25 route 100 cud ^pad$ interface serial 2
x25 route 100 interface serial 1
x25 route ^3306 interface serial 3
x25 route .* ip 10.2.0.2
```

The routing table forwards calls for X.121 address 100 out interface serial 0. Otherwise, calls are forwarded onto serial 1 if the X.121 address contains 100 anywhere within it and contains no call user data (CUD), or if the CUD is not the string *pad*. If the X.121 address contains the digits 100 and the CUD is the string *pad*, the call is forwarded onto serial 2. All X.121 addresses that do not match the first three routes are checked for a DNIC of 3306 as the first four digits. If they do match, they are forwarded over serial 3. All other X.121 addresses will match the fifth entry, which is a match-all pattern and will have a TCP connection established to the IP address 10.2.0.2. The router at 10.2.0.2 will then handle the call according to its configuration.

This second example configures a router that sits on a Tymnet/PAD switch to accept calls and have them forwarded to a DEC VAX system. This feature permits running an X.25 network over a generalized existing IP network, thereby making another physical line for one protocol unnecessary. The router positioned next to the DEC VAX system is configured with X.25 routes, as follows:

```
x25 route vax-x121-address interface serial 0
x25 route .* ip cisco-on-tymnet-ipaddress
```

These commands route all calls to the DEC VAX X.121 address out to serial 0, where the VAX is connected running PSI. All other X.121 addresses are forwarded to the *cisco-on-tymnet* address through its IP address. As a result, all outgoing calls from the VAX are sent to *cisco-on-tymnet* for further processing.

On the router named *cisco-on-tymnet*, you enter these commands:

```
x25 route vax-x121-address ip cisco-on-vax
x25 route .* interface serial 0
```

These commands force all calls with the VAX X.121 address to be sent to the router with the VAX connected to it. All other calls with X.121 addresses are forwarded out to Tymnet. If Tymnet can route them, a Call Accepted packet is returned, and everything proceeds normally. If Tymnet cannot handle the calls, it clears each call and the Clear Request packet is forwarded back toward the VAX.

PVC Used to Exchange IP Traffic Example

The following example, illustrated in Figure 29, demonstrates how to use the PVC to exchange IP traffic between Router X and Router Y.

Figure 29 Establishing an IP Encapsulation PVC through an X.25 Network

Configuration for Router X

```
interface serial 2
ip address 172.20.1.3 255.255.255.0
x25 pvc 4 ip 172.20.1.4
```

Configuration for Router Y

```
interface serial 3
ip address 172.20.1.4 255.255.255.0
x25 pvc 3 ip 172.20.1.3
```

In this example, the PDN has established a PVC through its network connecting PVC number 3 of access point A to PVC number 4 of access point B. On Router X, a connection is established between Router X and Router Y's IP address, 172.20.1.4. On Router Y, a connection is established between Router Y and Router X's IP address, 172.20.1.3.

Point-to-Point Subinterface Configuration Example

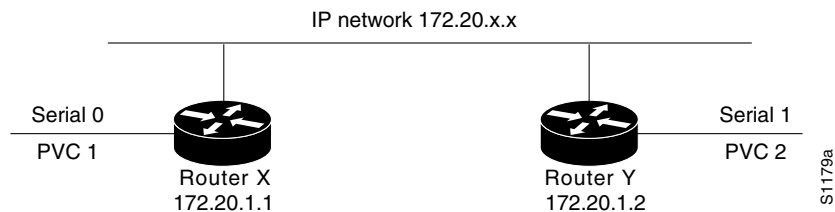
The following example creates a point-to-point subinterface, maps IP and AppleTalk to a remote host, and creates an encapsulating PVC for DECnet to the same remote host, identified by the X.121 address in the commands:

```
interface Serial0.1 point-to-point
  x25 map ip 172.20.170.90 170090 broadcast
  x25 map appletalk 4.50 170090 broadcast
  x25 pvc 1 decnet 1.2 170090 broadcast
```

Simple Switching of a PVC over XOT Example

In the following simple example, a connection is established between two PVCs across a LAN. Because the connection is remote (across the LAN), the XOT service is used. This example establishes a PVC between Router X, Serial 0, PVC 1 and Router Y, Serial 1, PVC 2. Keepalives are enabled to maintain connection notification. Figure 30 provides a visual representation of the configuration.

Figure 30 X.25 PVC Connection



Configuration for Router X

```
service tcp-keepalives-in
service tcp-keepalives-out
interface serial 0
  x25 pvc 1 xot 172.20.1.2 interface serial 1 pvc 2
```

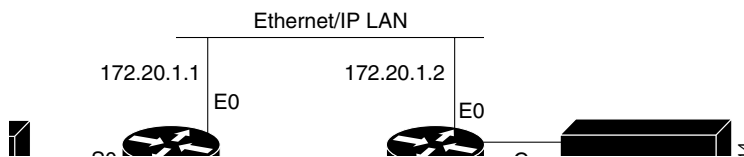
Configuration for Router Y

```
service tcp-keepalives-in
service tcp-keepalives-out
interface serial 1
  x25 pvc 2 xot 172.20.1.1 interface serial 0 pvc 1
```

PVC Switching over XOT Example

In the more complex example shown in Figure 31, the connection between points A and B is switched, and the connections between point C and points A and B are made using XOT. Keepalives are enabled to maintain connection notification.

Figure 31 PVC Switching over XOT



Configuration for Router X

```

service tcp-keepalives-in
service tcp-keepalives-out
interface ethernet 0
 ip address 172.20.1.1 255.255.255.0
!
interface serial 0
 x25 ltc 5
 x25 pvc 1 interface serial 1 pvc 1
 x25 pvc 2 xot 172.20.1.2 interface serial 0 pvc 1
!
interface serial 1
 x25 ltc 5
 x25 pvc 1 interface serial 0 pvc 1
 x25 pvc 2 xot 172.20.1.2 interface serial 0 pvc 2

```

Configuration for Router Y

```

service tcp-keepalives-in
service tcp-keepalives-out
interface ethernet 0
 ip address 172.20.1.2 255.255.255.0
!
interface serial 0
 x25 ltc 5
 x25 pvc 1 xot 172.20.1.1 interface serial 0 pvc 2
 x25 pvc 2 xot 172.20.1.1 interface serial 1 pvc 2

```

X.25 Switching between PVCs and SVCs Example

The following example allows X.25 switching between a PVC on the first interface and an SVC on the second interface. X.25 traffic arriving on PVC 20 on serial interface 0 will cause a call to be placed to 000000160100, if one does not already exist.

```
x25 routing
interface serial0
  encapsulation x25
  x25 address 000000180100
  x25 ltc 128
  x25 pvc 20 svc 000000160100 packetsize 128 128 windowsize 2 2

interface serial2
  encapsulation x25 dce
  x25 route ^000000160100$ interface Serial2
  x25 route ^000000180100$ interface Serial0
```

The **x25 route** command adds the two X.121 addresses to the X.25 routing table. Data traffic received on PVC 20 on serial interface 0 will cause a call to be placed with a Called (destination) Address of 000000160100; this call will be routed to serial interface 2. Alternatively, an X.25 call received with a Called Address of 000000180100 and a Calling Address of 000000160100 will be associated with PVC 20 on serial interface 0. In either case, subsequent X.25 traffic on either the SVC or the PVC will be forwarded to the other circuit. Because no idle timeout has been specified for the interface or for the circuit, the router will not clear the call.

CMNS Switching Example

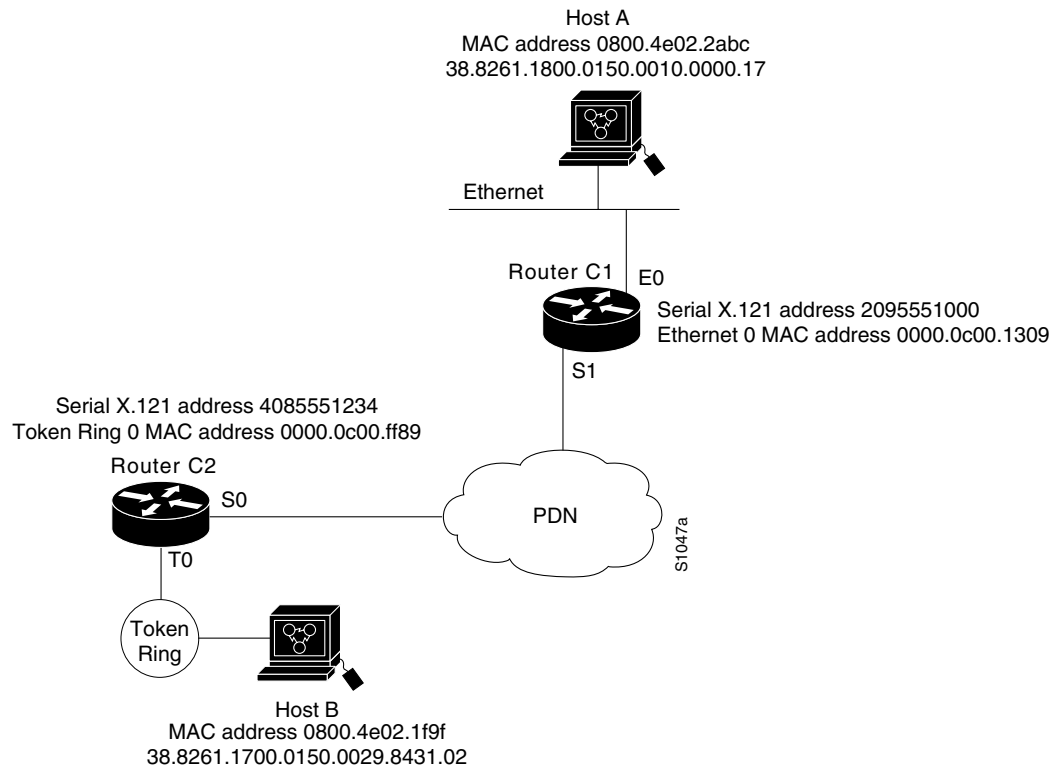
The following example illustrates enabling CMNS and configuring X.25 routes to the available CMNS host and the PDN connectivity:

```
interface ethernet 0
  cmns enable
!
interface serial 0
  encapsulation x25
!
interface serial 1
  encapsulation x25
!
x25 route dest-ext ^38.8261.1000.0150.1000.17 interface Ethernet0 mac 0000.0c00.ff89
! Above maps NSAP to MAC-address on Ethernet0
!
x25 route dest-ext ^38.8261.1000.0150.1000.18 substitute-dest 3110451 interface Serial0
! Above maps NSAP to X.121-address on Serial0 assuming the link is over a PDN
!
x25 route dest-ext ^38.8261.1000.0150.1000.20 interface Serial1
! Above specifies cmns support for Serial1
! assuming that the link is over a leased line
```

CMNS Switched over a PDN Example

The following example depicts switching CMNS over a packet-switched public data network (PDN). Figure 32 illustrates the general network topology for a CMNS switching application where calls are being made between resources on opposite sides of a remote link to Host A (on an Ethernet) and Host B (on a Token Ring), with a PDN providing the connection.

Figure 32 Example Network Topology for Switching CMNS over a PDN



The following configuration listing allows resources on either side of the PDN to call Host A or Host B. This configuration allows traffic intended for the remote NSAP address specified in the **x25 route** commands (for the serial ports) to be switched through the serial interface for which CMNS is configured.

Configuration for Router C2

```
interface token 0
  cmns enable
!
interface serial 0
  encapsulation x25
  x25 address 4085551234
!
x25 route dest-ext ^38.8261.17 interface Token0 mac 0800.4e02.1f9f
!
! The line above specifies that any traffic from any other interface
! intended for any NSAP address with NSAP prefix 38.8261.17 will be
! switched to MAC address 0800.4e02.1f9f through Token Ring 0
!
x25 route dest-ext ^38.8261.18 substitute-dest 2095551000 interface Serial0
!
! The line above specifies that traffic from any other interface
! on Cisco Router C2 that is intended for any NSAP address with
! NSAP-prefix 38.8261.18 will be switched to
! X.121 address 2095551000 through Serial 0
```

Configuration for Router C1

```

interface ethernet 0
  cmns enable
!
interface serial 1
  encapsulation x25
  x25 address 2095551000
!
x25 route dest-ext ^38.8261.18 interface Ethernet0 mac 0800.4e02.2abc
!
! The line above specifies that any traffic from any other
! interface intended for any NSAP address with NSAP 38.8261.18
! will be switched to MAC address 0800.4e02.2abc through Ethernet 0
!
x25 route dest-ext ^38.8261.17 substitute-dest 4085551234 interface Serial1
!
! The line above specifies that traffic from any other interface
! on Cisco Router C1 that is intended for any NSAP address with
! NSAP-prefix 38.8261.17 will be switched to X.121 address
! 4085551234 through Serial 1

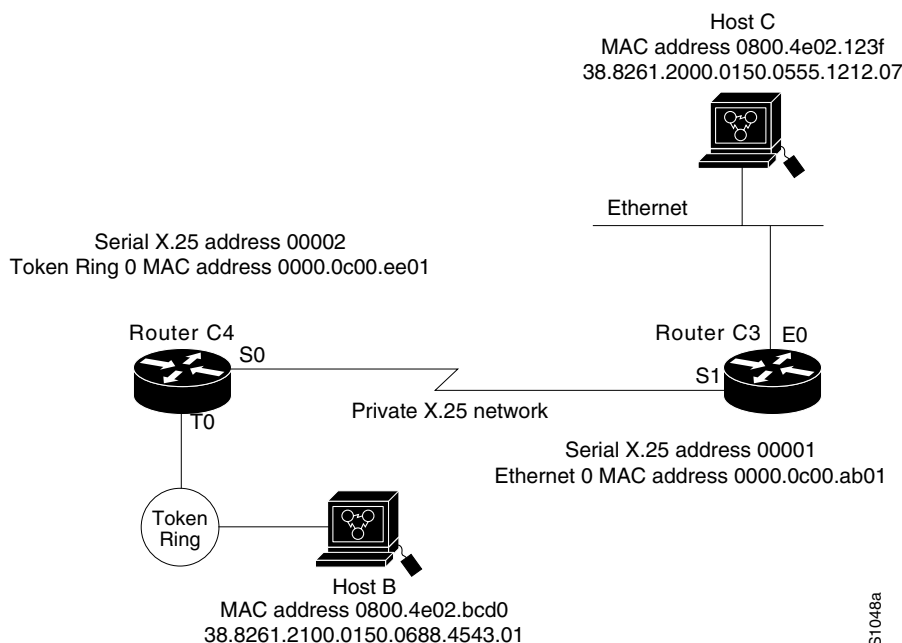
```

CMNS Switched over Leased Lines Example

The following example illustrates switching CMNS over a leased line. Figure 33 illustrates the general network topology for a CMNS switching application where calls are being made by resources on the opposite sides of a remote link to Host C (on an Ethernet) and Host B (on a Token Ring), with a dedicated leased line providing the connection.

The following configuration listing allows resources on either side of the leased line to call Host C or Host B. This configuration allows traffic intended for the remote NSAP address specified in the **x25 route** commands (for the serial ports) to be switched through the serial interface for which CMNS is configured.

Figure 33 Example Network Topology for Switching CMNS over a Leased Line



S1048a

A key difference for this configuration compared with the previous example is that with no PDN, the substitution of the destination X.121 address in the **x25 route** command is not necessary. The specification of an X.25 address also is not needed, but is included for symmetry with the previous example.

Configuration for Router C4

```
interface token 0
  cmns enable
!
interface serial 0
  encapsulation x25
  x25 address 4085551234
!
x25 route dest-ext ^38.8261.17 interface Token0 mac 0800.4e02.1f9f
!
! The line above specifies that any traffic from any other interface
! intended for any NSAP address with NSAP prefix 38.8261.17 will be
! switched to MAC address 0800.4e02.1f9f through Token Ring 0
!
x25 route dest-ext ^38.8261.18 interface Serial0
!
! The line above specifies that traffic from any other interface
! on Cisco Router C2 that is intended for any NSAP address with
! NSAP-prefix 38.8261.18 will be switched to
! X.121 address 2095551000 through Serial 0
```

Configuration for Router C3

```
interface ethernet 0
  cmns enable
!
interface serial 1
  encapsulation x25
  x25 address 2095551000
!
x25 route dest-ext ^38.8261.18 interface Ethernet0 mac 0800.4e02.2abc
!
! The line above specifies that any traffic from any other
! interface intended for any NSAP address with NSAP 38.8261.18
! will be switched to MAC address 0800.4e02.2abc through Ethernet 0
!
x25 route dest-ext ^38.8261.17 interface Serial1
!
! The line above specifies that traffic from any other interface
! on Cisco Router C1 that is intended for any NSAP address with
! NSAP-prefix 38.8261.17 will be switched to X.121 address
! 4085551234 through Serial 1
```

DDN X.25 Configuration Example

The following example illustrates how to configure a router interface to run DDN X.25:

```
interface serial 0
  ip address 192.31.7.50 255.255.255.240
  encapsulation x25 ddn
  x25 win 6
  x25 wout 6
  x25 ips 1024
  x25 ops 1024
  x25 t20 10
```

```
x25 t21 10
x25 t22 10
x25 t23 10
x25 nvc 2
x25 map IP 192.31.7.49 000000010300 BROADCAST
```

Blacker Emergency Mode Example

In the following example, interface serial 0 is configured to require an EXEC command from you or your network administrator before it participates in emergency mode. The host IP address is 21.0.0.12, and the address of the remote BFE unit is 21.0.0.1. When the BFE enters emergency mode, the router prompts for the EXEC command **bfe enter** to direct the router to participate in emergency mode.

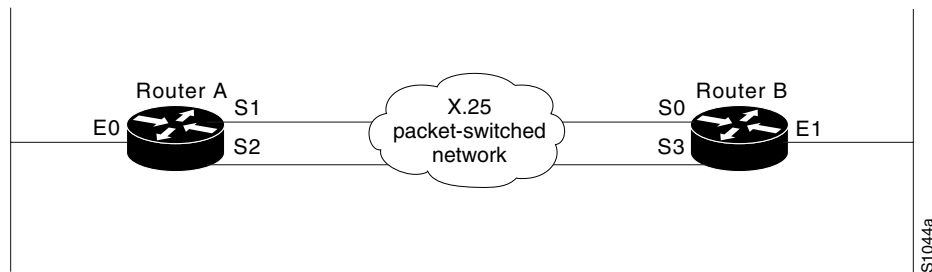
```
interface serial 0
 ip address 21.0.0.2 255.0.0.0
 encapsulation x25 bfe
 x25 bfe-emergency decision
 x25 remote-red 21.0.0.12 remote-black 21.0.0.1
 x25 bfe-decision ask
```

X.25 Configured to Allow Ping Support over Multiple Lines Example

For **ping** commands to work in an X.25 environment (when load sharing over multiple serial lines), you must include entries for all adjacent interface IP addresses in the **x25 map** command for each serial interface. The following example illustrates this point.

Consider two routers, Router A and Router B, communicating with each other over two serial lines via an X.25 PDN (see Figure 34) or over leased lines. In either case, all serial lines must be configured for the same IP subnet address space. The configuration that follows allows for successful **ping** commands. A similar configuration is required for the same subnet IP addresses to work across X.25.

Figure 34 Parallel Serial Lines to an X.25 Network



Note All four serial ports configured for the two routers in the following configuration example must be assigned to the same IP subnet address space. In this case, the subnet is 172.20.170.0.

Configuration for Router A

```
interface serial 1
 ip 172.20.170.1 255.255.255.0
 x25 address 31370054068
 x25 alias ^31370054069$
 x25 map ip 172.20.170.3 31370054065
 x25 map ip 172.20.170.4 31370054065
!
interface serial 2
 ip 172.20.170.2 255.255.255.0
 x25 address 31370054069
 x25 alias ^31370054068$
 x25 map ip 172.20.170.4 31370054067
 x25 map ip 171.20.170.3 31370054067
! allow either destination address
```

Configuration for Router B

```
interface serial 0
 ip 172.20.170.3 255.255.255.0
 x25 address 31370054065
 x25 alias ^31370054067$
 x25 map ip 172.20.170.1 31370054068
 x25 map ip 172.20.170.2 31370054068
!
interface serial 3
 ip 172.20.170.4 255.255.255.0
 x25 address 31370054067
 x25 alias ^31370054065$
 x25 map ip 172.20.170.2 31370054069
 x25 map ip 172.20.170.1 31370054069
! allow either destination address
```

Booting from a Network Server over X.25 Example

You cannot boot the router over an X.25 network using broadcasts. Instead, you must boot from a specific host. Also, an **x25 map** command must exist for the host that you boot from. The **x25 map** command is used to map an IP address into an X.121 address. There must be an **x25 map** command that matches the IP address given on the **boot system** command line. The following is an example of such a configuration:

```
boot system gs3-k.100 172.18.126.111
!
interface Serial 1
 ip address 172.18.126.200 255.255.255.0
 encapsulation X25
 x25 address 10004
 x25 map IP 172.18.126.111 10002 broadcast
 lapb n1 12040
 clockrate 56000
```

In this case, 10002 is the X.121 address of the remote router that can get to host 172.18.126.111.

The remote router must have the following **x25 map** entry:

```
x25 map IP 172.18.126.200 10004 broadcast
```

This entry allows the remote router to return a boot image from the host to the router booting over X.25.

X.29 Access List Example

The following example illustrates an X.29 access list. Incoming permit conditions are set for all IP hosts and LAT nodes that have specific characters in their names. All X.25 connections to a printer are denied. Outgoing connections are list restricted.

```
!Permit all IP hosts and LAT nodes beginning with "VMS".
!Deny X.25 connections to the printer on line 5.
!
access-list 1 permit 0.0.0.0 255.255.255.255
  lat access-list 1 permit ^VMS.*
  x29 access-list 1 deny .*
!
line vty 5
  access-class 1 in
!
!Permit outgoing connections for other lines.
!
!Permit IP access with the network 172.30
access-list 2 permit 172.30.0.0 0.0.255.255
!
!Permit LAT access to the boojum/snark complexes.
  lat access-list 2 permit ^boojum$
  lat access-list 2 permit ^snark$
!
!Permit X.25 connections to Infonet hosts only.
  x29 access-list 2 permit ^31370
!
line vty 0 16
  access-class 2 out
```

X.29 Profile Script Example

The following profile script turns local edit mode on when the connection is made and establishes local echo and line termination upon receipt of a Return. The name *linemode* is used with the **translate** command to effect use of this script.

```
x29 profile linemode 2:1 3:2 15:1
translate tcp 172.30.1.26 x25 55551234 profile linemode
```

The X.3 PAD parameters set in the profile file and the **translate** command are described in the “Configuring Protocol Translation” chapter in *Dial Solutions Configuration Guide*.