Chapter 1 *Configuring Packet-Switched Software*

1

This chapter describes the configuration tasks for the Cisco packet-switched software. These tasks include configuring the following protocols and services:

- Recommendation X.25, including Link Access Procedure, Balanced (LAPB)
- Frame relay service
- Switched Multi-Megabit Data Services (SMDS)

Summaries of the commands to configure and maintain each service are listed at the end of each section.

Configuring LAPB

It is possible to only use LAPB as a serial encapsulation method. This can be done using a leased serial line. You must use one of the X.25 packet-level encapsulations when attaching to an X.25 network.

Using LAPB under noisy conditions can result in greater throughput than HDLC encapsulation. When LAPB detects a damaged frame, the router immediately retransmits the frame instead of waiting for host timers to expire. 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 retransmitting host retransmissions.

However, if the line is not noisy, the lower overhead of HDLC encapsulation is more efficient than LAPB. When using long delay satellite links, the lock step behavior of LAPB makes the use of HDLC encapsulation the better choice.

The X.25 Recommendation distinguishes between two types of X.25 hosts: data terminal equipment (DTE) in LAPB encapsulation hosts, and data communications equipment (DCE) in LAPB encapsulation hosts.

A router using LAPB encapsulation can act as a DTE or DCE device at the protocol level.

Running a Single Network Protocol

To run datagrams of a single protocol over a serial interface using the LAPB encapsulation, use the interface subcommand:

encapsulation {**lapb**|**lapb-dce**}

The keyword **lapb** sets DTE operation; the keyword **lapb-dce** sets DCE operation. One end of the link must be DTE and the other must be DCE. By default, the single protocol is IP.

To configure another protocol, use the interface subcommand:

lapb protocol *keyword*

Possible protocol keywords include **ip**, **xns**, **decnet**, **appletalk**, **vines**, **clns**, **novell**, and **apollo**.

Running Multiple Network Protocols

To enable use of multiple network protocols on the same line at the same time, use the keyword **multi-lapb** or **multi-lapb-dce** for DTE or DCE operation, respectively:

encapsulation {**multi-lapb**|**multi-lapb-dce**}

For example, with the **multi-lapb** or **multi-lapb-dce** keyword, you can use IP, DECnet, and XNS at the same time. Both ends of the line must use the same encapsulation: either **lapb** or **multi-lapb**. One end of each line must be DCE.

Sample Configuration of LAPB Encapsulation

In the following example of LAPB encapsulation configuration, the frame size (N1), window size (K), hold timer (TH), and maximum retransmission (N2) parameters retain their default values. The **encapsulation** subcommand sets DCE operation for IP packets only, and the **lapb t1** subcommand sets the retransmission timer to 4,000 milliseconds (4 seconds).

Example:

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

For more information on LAPB parameters, see the next section, “Setting the X.25 Level 2 (LAPB) Parameters.”

Setting the X.25 Level 2 (LAPB) Parameters

X.25 Level 2, or LAPB (Link Access Procedure, Balanced), is a data encapsulation protocol that operates at Level 2 (the data link level) 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.

LAPB parameters are set with the **lapb** interface subcommand. The interface must be running with either a LAPB or X.25 encapsulation method specified by the **encapsulation** interface subcommand. This subcommand takes two required arguments, *parameter* and *value*. The argument *parameter* is one of several keywords described in the following text, and the argument *value* is a decimal number representing a period of time, a bit count, or a frame count, depending on parameter. Table 1-1 summarizes the LAPB parameters.

Table 1-1 LAPB Parameters

Parameter	Value	Value Range	Default
k	*frames*	1-7	7
n1	*bits*	1-16384	12000
n2	*times*	1-255	20
t1	*milliseconds*	1-64000	3000

S1613

Note: The LAPB TH value is not included as a configurable parameter; the value is always 0.

Setting the Retransmission Timer

The retransmission timer determines how long a transmitted frame can remain unacknowledged before the router polls for an acknowledgment. To set the limit for the retransmission timer (the LAPB T1 parameter), use this subcommand:

lapb t1 *milliseconds*

The argument *milliseconds* is the number of milliseconds from 1 through 64000. The default value is 3,000 milliseconds.

For X.25 networks, the router retransmission timer setting should match that of the network. Mismatched retransmission timers can cause excessive retransmissions and an effective loss of bandwidth.

For leased-line circuits, the retransmission timer setting is critical. The timer setting must be large enough to permit several maximum-sized frames to complete one round trip on the link. If the timer setting is too small, the router will poll before the acknowledgment frame can return, which results in an effective loss of bandwidth. If the timer setting is too large, the router waits longer than necessary before requesting an acknowledgment, which also reduces bandwidth.

To determine an optimal value for the retransmission timer, use the privileged EXEC command **ping** to measure the round-trip time of a maximum-sized frame on the link. Multiply this time by a safety factor that takes into account the speed of the link, the link quality, and the distance. A typical safety factor is four. Choosing a larger safety factor can result in slower data transfer if the line is noisy. However, this disadvantage is minor compared to the excessive retransmissions and effective bandwidth reduction caused by a timer setting that is too small.

Setting Frame Parameters

To specify the maximum number of bits a frame can hold, use the **lapb n1** interface subcommand:

lapb n1 *bits*

The **n1** keyword specifies the maximum number of bits (N1) a frame can hold. The argument *bits* is the number of bits from 1 through 16,384, and must be a multiple of eight. The default value is 12,000 bits (1500 bytes).

When connecting to an X.25 network, use the N1 parameter value set by the network administration, which is the maximum size of an X.25 packet. When using LAPB over leased lines, the N1 parameter should be eight times the MTU.

To specify the maximum number of times an acknowledgment frame can be retransmitted, use the **lapb n2** interface subcommand:

lapb n2 *retries*

The argument *retries* is the retransmission count from 1 through 255. The default value is 20 retransmissions.

To specify the maximum permissible number of outstanding frames, called the *window size*, use the **lapb k** interface subcommand:

lapb k *window-size*

The argument *window-size* is a packet count from one to seven. The default value is seven packets.

Monitoring and Troubleshooting LAPB

To display operation statistics for an interface using LAPB encapsulation, use the EXEC command **show interfaces.**

The following example output shows the state of the LAPB protocol, the current parameter settings, and a count of the different types of frames. Each frame count is displayed in the form sent/received.

```
LAPB state is DISCONNECT, T1 3000, N1 12000, N2 20, K 7,TH 3000
IFRAMEs 12/28 RNRs 0/1 REJs 13/1 SABMs 1/13 FRMRs 3/0 DISCs 0/11
```

For a description of the variable names in the **show interface** output, see the X.25 recommendation.

To debug LAPB problems, you must have a good understanding of the X.25 recommendation.

To enable the logging of all packets received and generated, use the privileged EXEC command **debug lapb**. Note that this command slows down processing considerably on heavily loaded links. The following shows example output:

```
Serial0: LAPB O CONNECT (5) IFRAME 0 1
Serial0: LAPB I CONNECT (2) RR 1 (R)
Serial0: LAPB I CONNECT (5) IFRAME P 2 1 (C)
Serial0: LAPB O REJSENT (2) REJ P/F 1
Serial0: LAPB I REJSENT (2) DM F (C)
Serial0: LAPB I DISCONNECT (2) SABM (C)
Serial0: LAPB O CONNECT (2) UA
.
.
.
Serial0: LAPB T SABMSENT 357964 0
Serial0: LAPB O SABMSENT (2) SABM P
```

In the example output, each line represents a LAPB frame entering or exiting the router. The first field shows the interface type and unit number of the interface reporting the frame event. The second field is the protocol that provided the information.

The third field is I, O, or T for "frame input," "frame output," or "T1 timer expired," respectively. The fourth field indicates the state of the protocol when the frame event occurred. In a timer event, the state name is followed by the current timer value and the number of retransmissions.

In a packet input or output event, the state name is followed by the size of the frame in bytes (in parentheses) and the frame type name. The next field is an optional indicator: P/F, P, or F, which stand for "Poll/Final," "Poll," and "Final," respectively. For IFRAME frames only, the next two numbers are the receive and send sequence numbers, respectively. For RR, RNR, and REJ frames, the next number is the receive sequence number. For FRMR frames, the next three numbers are three bytes of error data. The last optional indicator is (C) or (R) for "command" or "response," respectively.

Configuring X.25

The software for the Cisco network server products supports the 1980 and 1984 Recommendation X.25 published by the French International Telegraph and Telephone Consultative Committee (CCITT). The Recommendations specify connections between data terminal equipment (DTE) and data communications equipment (DCE). The Defense Data Network (DDN) and the International Standards Organization (ISO) specify the use of X.25 protocol for computer communications. Many public and private networks also use Recommendation X.25 as their interface technology.

The X.25 model is a telephone network for computer data communications. To start data communications, one computer system calls another to request a communications session. The called computer system can accept or refuse the call. If the called system accepts the call, the two computer systems can begin transferring data in both directions; either system can terminate the call.

In addition to providing remote terminal access, X.25 networks provide bridging capability using a growing list of protocols—the Internet Protocol (IP), DECnet, XNS, ISO CLNS, AppleTalk, Novell IPX, Banyan VINES, and Apollo Domain.

The following sections provide an overview of the Cisco X.25 implementation, describing the different encapsulation methods supported by X.25, and X.25 as a datagram transport, with special attention to the IP protocol. This is followed by descriptions of the DDN X.25 support provided by the Cisco routers, and descriptions of how to configure X.25 switching and bridging. Finally, the configuration of the X.25 Level 3 parameters, and the X.25 Level 2 facilities are discussed. A summary of the interface subcommands are provided at the end of the section.

Note: The default values provided by the software are sufficient for most X.25 networks; however, some parameters may need to be configured, depending on the network.

Overview of Cisco X.25 Support

Cisco Systems' X.25 support can be used in two different ways:

- As a transport for datagram traffic—This entails encapsulating datagrams of IP, DECnet, AppleTalk, and so forth inside packets on an X.25 virtual circuit. Mappings between X.25 addresses and protocol addresses allow these datagrams to be routed through an X.25 network.
- As an X.25 switch—X.25 calls can be routed based on their X.25 addresses either between serial interfaces on the same router (local switching) or across an IP network to another Cisco router (remote switching).

Remote X.25 switching encapsulates the X.25 packet-level inside a TCP connection, allowing disjoint X.25 networks to be connected via a TCP/IP-based network.

X.25 Encapsulation Methods

This section describes the different encapsulation methods and commands that Cisco supports for commercial and private X.25 networks.

Methods of encapsulation for DDN networks are described in the section "Configuring the Datagram Transport on DDN Networks" later in this chapter.

Configuring X.25 DTE and DCE Operation

A router using X.25 Level 3 encapsulation can act as a DTE or DCE device on general X.25 networks.

To set X.25 DTE operation, use the **encapsulation x25** interface subcommand:

encapsulation x25

To set X.25 DCE operation, use the **encapsulation x25-dce** interface subcommand:

encapsulation x25-dce

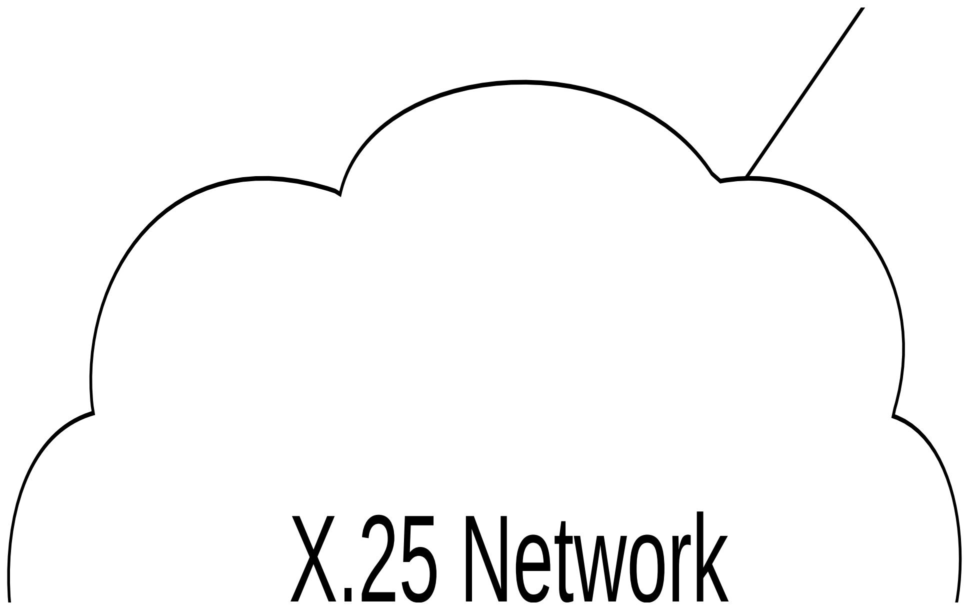
Configuring the Datagram Transport on Commercial X.25 Networks

Cisco Systems' X.25 support is most commonly configured as a transport for datagrams across an X.25 network. This is accomplished by first establishing a mapping between protocol addresses (for example, IP or DECnet) and the X.121 addresses of the X.25 network. When datagrams for a particular destination are routed for the first time, a virtual circuit is set up to the appropriate X.121 address. The Call User Data portion of the initial Call Request identifies the protocol of the datagrams being carried by a particular virtual circuit. If multiple protocols are in use, multiple virtual circuits will be opened.

Address Mapping Issues

To transport datagrams using X.25 Level 3, the router must map network-protocol addresses to X.121 addresses and vice versa. For example, Figure 1-1 illustrates Hosts A and B that want to communicate via Routers X and Y, which have an X.25 link between them.

Figure 1-1 Communicating via Routers Through an X.25 Network



To send a packet to Host B, Host A must first send the packet to Router X. From the destination address in the packet and from its routing information, Router X determines that it must send the packet to Router Y over the X.25 network. Router X must then determine the X.121 address for Router Y to open a virtual circuit. Router X uses its network-protocol-to-X.121 address map to convert the protocol address of Router Y to the equivalent

X.121 address. Router X can now make the call to create a virtual circuit.

Except in the case of IP using DDN X.25 encapsulation, there is no standard method for dynamically determining these mappings. Instead, static mapping tables must be configured into each router.

To display the network-protocol-to-X.121 address mapping, use the EXEC command **show x25 map.**

Setting Address Mappings

To specify a network-protocol-to-X.121 address mapping such as Internet-to-X.121 or DECnet-to-X.121, use the **x25 map** interface subcommand:

x25 map *protocol-keyword protocol-address X.121-address* [*option1*... *option6*]
no x25 map *protocol-keyword protocol-address X.121-address*

Note: For bridging over X.25, there is no protocol address; however, the broadcast option is required.

The argument *protocol-keyword* is one of these keywords:

- **ip**—IP
- **clns**—ISO CLNS
- **decnet**—DECnet
- **chaos**—CHAOSnet
- **xns**—XNS
- **novell**—Novell IPX
- **appletalk**—AppleTalk
- **vines**—VINES
- **apollo**—Apollo Domain
- **pup**—PUP

The *address* arguments specify the network-protocol-to-X.121 mapping.

The *option* arguments add certain features to the mapping specified, and can be any of the following, up to six. They must be specified in the order listed.

- **reverse**—Specifies reverse charging for outgoing calls.
- **accept-reverse**—Causes the router to accept incoming reverse-charged calls. If this option is not present, the router clears reverse charge calls.
- **broadcast**—Causes the router to direct any broadcasts sent through this interface to the specified X.121 address. This option is needed when dynamic routing protocols are

being used to access the X.25 network, and is required for bridging X.25.

- **cug** *number*—Specifies a Closed User Group number (from 1 to 99) for the mapping in the outgoing call.
- **nvc** *count*—Sets the number of virtual circuits (VCs) for this protocol/host. The default *count* is the **x25 nvc** setting of the interface. A maximum number of eight VCs can be configured for a single protocol/host.
- **packetsize** *in-size out-size*—Specifies input packet size *in-size* and output packet size *out-size* for the mapping in the outgoing call.
- **windowsize** *in-size out-size*—Specifies input window size *in-size* and output window size *out-size* for the mapping in the outgoing call.
- **throughput** *in out*—Requests the amount of bandwidth through the X.25 network.
- **modulo** *size*—Specifies the maximum window size for this map. The argument *size* permits windows of 8 or 128 on the same interface.
- **nuid** *username password*—Specifies that a network ID facility be sent in the outgoing call with the specified user name and password.

To retract a network-protocol-to-X.121 mapping, use the **no x25 map** interface subcommand with the appropriate network protocol and X.121 address arguments.

Bridging on X.25

Cisco's transparent bridging software supports bridging of X.25 frames. To configure this capability, add this variation of the **x25 map** interface subcommand to the bridging configuration file:

x25 map bridge *X.121-address* **broadcast** [*options-keywords*]

The command specifies Internet-to-X.121 address mapping. The keyword **bridge** specifies bridging over X.25. The argument *X.121-address* is the X.121 address. The keyword **broadcast** is required for bridging X.25 frames. The argument *options-keywords* are the are services that may be added to this map as listed in the section "Setting Address Mappings" earlier in this chapter. For further information about bridging on X.25, and for an example configuration, refer to the chapter "Configuring Transparent Bridging."

Configuring the X.25 Parameters

The Cisco router software provides subcommands to configure the standard Level 2 and Level 3 X.25 parameters and user facilities.

This section discuses the commands and procedures needed to set these parameters, including interface addresses, virtual circuit channel sequence, and addresses.

Setting Permanent Virtual Circuits

Permanent Virtual Circuits (PVCs) are the X.25 equivalent of leased lines; they are never disconnected. To establish a PVC, use the **x25 pvc** interface subcommand:

x25 pvc *circuit protocol-keyword protocol-address*
no x25 pvc *circuit protocol-keyword protocol-address*

The argument *circuit* is a virtual-circuit channel number and it must be less than the lower limit of the incoming call range in the virtual circuit channel sequence.

The argument *protocol-keyword* can be one of these keywords:

- **ip**—IP
- **clns**—ISO CLNS
- **decnet**—DECnet
- **chaos**—CHAOSnet
- **xns**—XNS
- **novell**—Novell IPX
- **appletalk**—AppleTalk
- **vines**—VINES
- **apollo**—Apollo Domain
- **pup**—PUP
- **bridged**—Bridged

The argument *protocol-address* is that of the host at the other end of the PVC.

To delete a PVC, use the **no x25 pvc** interface subcommand with the appropriate channel number, protocol keyword, and protocol address.

Note: You must specify the required network-protocol-to-X.121 address mapping with an **x25 map** subcommand before you can set up a PVC.

Note: When you are configuring X.25 to use a PVC, you must ensure that no traffic is sent towards the remote router between the time the X.25 map command is issued and the time that the X.25 PVC command is issued. Otherwise, the local router will create an SVC, and then the PVC command will not be allowed.

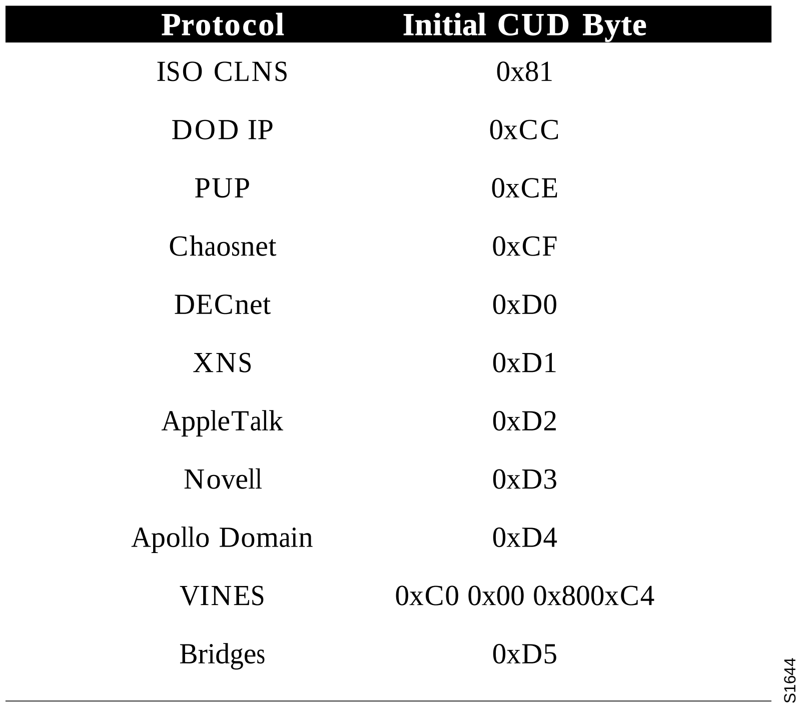
Note: Map entries with Broadcast attributes are particularly likely to get traffic. The simplest way to ensure that no traffic is sent while configuring is to shut down the interface while configuring it for PVC.

Protocol-to-Virtual-Circuit Mapping

The Call Request packet that sets up a virtual circuit contains a field called the Call User Data field. Typically, the first byte of Call User Data is used by Cisco software to distinguish which high-level protocol will be carried by a particular virtual circuit.

Table 1-2 lists the hexadecimal values of the initial byte of Call User Data and its corresponding network level protocol. The use of 0x81 for ISO 8473 (CLNS) is an ISO standard. The use of 0xCC for Department of Defense IP is defined by RFC 877. The use of C0 00 80 C4 is defined by Banyan. The other values are meaningful only to Cisco software. All the values are padded with three bytes of 0x00, except for VINES, the BFE X.25 encapsulation and CLNS, which follow ISO 8473 requirements.

Table 1-2 Protocols and Initial Byte of Call User Data



Protocol	Initial CUD Byte
ISO CLNS	0x81
DOD IP	0xCC
PUP	0xCE
Chaosnet	0xCF
DECnet	0xD0
XNS	0xD1
AppleTalk	0xD2
Novell	0xD3
Apollo Domain	0xD4
VINES	0xC0 0x00 0x800xC4
Bridges	0xD5

S1644

To set the default protocol, use the x25 default command. The full syntax follows:

x25 default *protocol*
no x25 default *protocol*

The **x25 default** command specifies the protocol assumed by the router to interpret calls with unknown Call User Data. The argument *protocol* sets the default protocol and is either **ip** or **pad**. Use this subcommand to change the action taken when an incoming call is received without identifying Call User Data. Normally, the call is cleared. When you use this subcommand, the incoming call is assumed to contain the specified default protocol.

The **no x25 default** subcommand removes the protocol specified.

Displaying Address Mappings

To display the network-protocol-to-X.121 address mappings, enter this command at the EXEC prompt:

show x25 map

The following is a sample output:

```
Serial0: novel 10.0.0c00.7b22 000000220200 PERMANENT, 1 LCN: 4*
Serial0: IP 10.1.0.1 000000010100 CONSTRUCTED
Serial1: appletalk 128.1 000000010000 PERMANENT
Serial1: decnet 28.1 000000020000 PERMANENT BROADCAST
Serial1: ip 128.1.0.3 000000030000 PERMANENT, 2 LCN: 1023*, 1024
```

Each line of output shows the interface name, the protocol type, the protocol address, the X.121 address, and the address-mapping type. The address-mapping types are `PERMANENT`, entered with the **x25 map** interface subcommand, `INTERFACE`, indicating the address of a router network interface, and `CONSTRUCTED` (derived using the DDN address conversion scheme).

If broadcasts are enabled for an address mapping, the word `BROADCAST` also appears on the output line. Finally, each line also shows the number of Logical Circuit Numbers (LCNs) and, if greater than zero, a list of the LCNs for the protocol/X.121 address. An asterisk marks the current LCN.

Example X.25 Configuration

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 *131.108.9.0* has been assigned for the X.25 network.

Note: When routing IP over X.25, the X.25 network must be treated 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 should have map entries for all others. There are also issues with the broadcast flag, which apply both to IP and to other protocols with dynamic routing.

```
interface serial 2
ip address 131.108.9.1 255.255.255.0
!
encapsulation X25
!
! The ”bandwidth” command is not part of the X.25
! configuration; it's especially important to understand that it doesn't
! have any connection with the X.25 entity of the same name. cisco
!”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, and X.25 service at that rate isn't 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
!
! These Level 3 parameters are defaults; they need to
! match PDN defaults. They are negotiable on a per-call basis:
!
x25 win 7
x25 wout 7
x25 ips 512
x25 ops 512
!
! You must specify an X.25 address, which you get from 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 may not need to change any Level 2
! parameters, however.
!
x25 hic 32
x25 htc 32
x25 hoc 32
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 them, 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 131.108.9.3 31370019134 ACCEPT-REVERSE
! (ACCEPT-REVERSE allows collect calls)
x25 map IP 131.108.9.1 31370054065
x25 map IP 131.108.9.2 31370053087
!
! The PDN cannot handle fast back-to-back packets, so the
!”transmitter-delay” command is used to slow down the MCI card:
!
transmitter-delay 1000
```

Configuring the Datagram Transport on DDN Networks

The DDN X.25 protocol has two versions: Basic Service and Standard Service. Using the DDN X.25 Basic Service, network devices send raw X.25 data across the DDN and assume no structure within the data portion of the X.25 packet. Basic Service users can interoperate only with other Basic Service users.

DDN X.25 Standard Service requires that the X.25 data packets carry IP datagrams. Because the DDN Packet Switch Nodes (PSNs) can extract the Internet packet from within the X.25 packet, they can pass data to either an 1822-speaking-host or to another Standard Service host. Thus, hosts using the older 1822 network interface can interoperate with hosts using Standard Service.

The DDN X.25 Standard is the required protocol for use with DDN PSNs. The Defense Communications Agency (DCA) has certified the Cisco Systems DDN X.25 Standard implementation for attachment to the Defense Data Network.

Enabling DDN X.25

A router using the DDN X.25 Basic Service can act as a DTE or DCE device. To set operation type, use the **encapsulation** interface subcommand:

encapsulation {**ddnx25**|**ddnx25-dce**}

These keywords cause the router to specify the Standard Service facility in the Call Request packet, which notifies the PSNs to use Standard Service.

Using Standard Service, the DDN can provide better service for virtual circuits with higher precedence values. If the router receives an Internet packet with a nonzero Internet precedence field, it opens a new virtual circuit and sets the precedence facility request to the DDN-specified precedence mapping in the Call Request packet. Different virtual circuits are maintained based on the precedence mapping values and the permitted number of virtual circuits.

For situations requiring a high degree of security, the Defense Data Network Blacker Front End Encryption (BFE) device is supported. If the router is attached to such a device, the **bfex25** keyword must be used with the **encapsulation** subcommand:

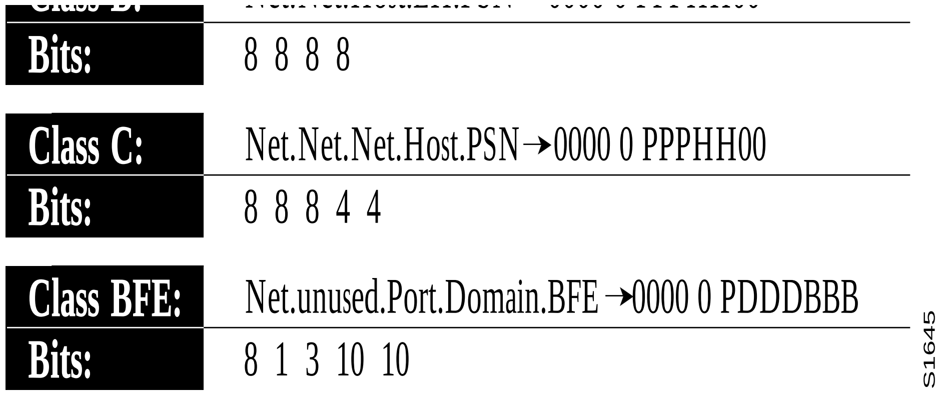
encapsulation bfex25

This encapsulation provides a mapping from Class A IP addresses to the type of X.121 addresses expected by the BFE encryption device.

DDN X.25 Dynamic Mapping

The DDN X.25 standard implementation includes a scheme for dynamically mapping all classes of Internet addresses to X.121 addresses without a table. This scheme requires that the Internet addresses conform to the formats shown in Table 1-3. These formats segment the Internet addresses into network (N), host (H), logical address (L), and IMP (I) portions. (The acronym IMP, which stands for Interface Message Processor, is the predecessor of PSN, which stands for Packet Switch Node.) For the BFE encapsulation, the Internet address is segmented into Port (P), Domain (D), and BFE ID number (B).

Table 1-3 DDN Internet/X.121 Address Conventions



[illegible]	[illegible]
Bits:	8 8 8 8
Class C:	Net.Net.Net.Host.PSN→0000 0 PPPHH00
Bits:	8 8 8 4 4
Class BFE:	Net.unused.Port.Domain.BFE→0000 0 PDDDBBB
Bits:	8 1 3 10 10

S1645

The DDN conversion scheme uses the host and IMP portions of an Internet address to create the corresponding X.121 address. Strictly speaking, the DDN conversion mechanism is limited to Class A Internet addresses. However, the router can convert Class B and Class C addresses as well. As indicated in Table 1-3, this method uses the last two octets of a Class B address as the host and IMP identifiers, and the upper and lower four bits in the last octet of a Class C address as the host and IMP identifiers, respectively.

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 ZZZZFIIIHHZZ(SS), where each character represents a digit. ZZZZ represents four zeros, F is zero to indicate a physical address, III represents the IMP (PSN) octet from the Internet address padded with leading zeros, HH is the host octet from the Internet address padded with leading zeros, and 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. If you do not specify the address for an interface configured for DDN X.25, the router uses the DDN mapping technique to obtain the X.121 address of an interface.

To display the network-protocol-to-X.121 address mapping, enter this command at the EXEC prompt:

show x25 map

This command is described in more detail later in this chapter. For more information on the BFE addressing, refer to the document *Blacker Interface Control Document (ICD)*, available by copying the file *blacker.doc* from Cisco's *ftp.cisco.com* public directory.

DDN X.25 Configuration Subcommands

Normally the X.25 parameters of a DDN connection are configured using the **x25** and **lapb** interface subcommand described in the section "Configuring the X.25 Parameters," earlier in this chapter. There are a few DDN-specific subcommands, however.

The Cisco X.25 implementation allows you to enable or disable the ability to open a new virtual circuit based on the IP Type of Service (TOS) field.

To do this, use the **x25 ip-precedence** interface subcommand. The full syntax of this command follows:

x25 ip-precedence
no x25 ip-precedence

By default, Cisco routers open one virtual circuit for each type of service. There is a problem associated with this in that some hosts send nonstandard data in the TOS field, thus causing multiple, wasteful virtual circuits to be created. The command **no x25 ip-precedence** causes the TOS field to be ignored when opening virtual circuits.

Using the HDH Protocol

As mentioned earlier, the HDH protocol (also known as the HDLC Distant Host or 1822-J protocol) provides a method for running the 1822 protocol over synchronous serial lines instead of over special-purpose 1822 hardware. HDH packets consist of 1822-LH/DH leaders and data encapsulated in LAPB (X.25 Level 2) format. The HDH hardware is on the Multiport Communications Interface (MCI) card. Strictly speaking, HDH is not X.25, however, it is still commonly used for attaching to the Defense Data Network.

The router supports both the packet and message modes of HDH. It is enabled with the **hdh** interface subcommand with the appropriate keyword. The syntax follows:

hdh {**packet**|**message**}

The **packet** keyword specifies the packet mode; the **message** keyword specifies the message mode.

To enable the HDH protocol, use the interface subcommand:

encapsulation hdh

To enable logging of HDH transactions, use the privileged EXEC command **debug hdh.** To enable logging of the underlying LAPB protocol transactions, use the privileged EXEC command **debug lapb.**

To display information about HDH, use the EXEC command **show imp-hosts.** This command displays information about the hosts with which the network server has communication during the past five minutes via its CSC-A (1822-LH/DH) interfaces or serial interface using HDH (1822-J) encapsulation.

Use the EXEC command **debug psn** to log information about the Packet Switch Node. This command enables logging of noteworthy Packet Switch Node (PSN) events for DDN network servers that are equipped with CSA-A (1822-LH/DH) interfaces or serial interfaces using HDH (1822-J) encapsulation.

The **debug psn-events** EXEC command enables logging of a subset of the PSN and 1822 debugging messages.

Configuring X.25 Switching

In addition to transporting datagrams, the Cisco X.25 software implementation allows switched virtual circuits to be forwarded from one X.25 interface to another and from one Cisco router to another. The forwarding behavior can be controlled based on a locally built table.

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

- The D-bit ignored but passed through transparently
- Variable length interrupt data
- Flow Control Parameter Negotiation
  - Window size up to 7
  - Packet size up to 2048
- The basic Closed User Group
- Throughput class negotiation
- Reverse charging and fast select
- Local facilities are stripped

Higher-level protocols may share an X.25 encapsulated serial interface with the X.25 switching support. The ability to switch or forward X.25 virtual circuits can be done in two different 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*, as the complete path is handled by the router itself. It does not matter whether the interfaces are configured as DTE or DCE, since software will take the appropriate actions.
- An incoming call can also be forwarded to another Cisco router using the TCP/IP protocols. Upon receipt of an incoming call, a TCP stream connection will be established to the Cisco router which is acting as the switch for the destination. All X.25 packets will be sent and received over this reliable data stream. Flow control is maintained from local DTE to remote DTE. This is known as *remote X.25 switching*.

Running X.25 over TCP/IP provides a number of benefits. The IP datagram containing the X.25 packet can be switched by other routers using the Cisco high-speed switching abilities. It also allows X.25 connections to 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 Cisco router where it can be output to an X.25 interface.

Enabling X.25 Switching

To enable X.25 switching, use the **x25 routing** global configuration command. The full syntax of this command follows:

x25 routing
no x25 routing

X.25 calls will not be forwarded until this command is issued. The command **no x25 routing** disables the forwarding of X.25 calls.

Constructing the X.25 Routing Table

The X.25 routing table is consulted when an incoming call is received that should be forwarded. The called (destination) X.121 address and Call User Data fields of the X.25 packet are used to determine the route.

An entry in the X.25 routing table is set up or removed with the **x25 rout**e global configuration commands. The full syntax and variations of these commands follows:

x25 route [**#** *position*]*x121-pattern* [**cud** *pattern*] **interface** *interface-name*
no x25 route [**#** *position*]*x121-pattern* [**cud** *pattern*] **interface** *interface-name*

x25 route [**#** *position*]*x121-pattern* [**cud** *pattern*] **ip** *ip-address*
no x25 route [**#** *position*]*x121-pattern* [**cud** *pattern*] **ip** *ip-address*

x25 route [**#** *position*]*x121-pattern* [**cud** *pattern*] **alias** *interface-name*
no x25 route [**#** *position*]*x121-pattern* [**cud** *pattern*] **alias** *interface-name*

The order in which X.25 routing table entries are specified is significant; the list is scanned linearly for the first match. The optional positional parameter (# followed by an integer) can be used to designate after which existing entry to insert or delete the new entry. If no positional parameter is given, the entry is appended to the end of the routing table.

The argument *pattern* is a regular expression that must match the called address and is required. Optional Call User Data corresponding to that X.121 address can be specified as a printable ASCII string. Both the X.121 address and Call User Data can be written using UNIX-style regular expressions. The Call User Data field specifies the data after the protocol identification field, which is four bytes. Use the **show x25 route** command to display the X.25 routing table.

The **alias** keyword permits a way for other X.121 addresses to be treated as local. An alias route is valid only for calls that come in on the named interface.

Enter the **no x25 route** command with the appropriate arguments and keywords to remove the entry from the table.

Translating X.25 Called Addresses

When interconnecting two separate X.25 networks, it is sometimes necessary to provide for address translation. Cisco's X.25 switch supports translation of X.25 called and calling addresses using the **substitute-dest** or **substitute-source** keyword with the **x25 route** configuration subcommand. Addresses can be rewritten using regular expression replacement.

The **substitute-source** keyword allows substitution of the calling address. For backwards compatibility, the **substitute** keyword will be accepted as **substitute-dest** and written to nonvolatile memory in the new format. When used with the **x25 use-source-address** command, this option allows the calling address to be modified.

x25 route [*#position*]*x121-pattern* [**substitute-source** *rewrite-pattern*] [**substitute-dest** *rewrite-pattern*] [**cud** *pattern*] *interface destination*

For typographical reasons, this command is shown on two lines. When using the optional keywords in this variation of the **x25 route** subcommand, the **substitute-source** keyword must precede the **substitute-dest** keyword, which both must precede the **cud** keyword, and the entire command must be on one line.

The argument *rewrite-pattern* will replace the called or calling X.121 address in routed X.25 packets, as appropriate. The backslash (\) character is treated specially in the argument *rewrite-pattern*; it indicates that the digit immediately following it selects a portion of the original called address to be inserted in the new called address. The characters \0 are replaced with the entire original address. The characters \1 through \9 are replaced with the strings that matched the first through ninth parenthesized parts of *X121-pattern*. See Table 1-4 for a summary of pattern and character matching. A more complete description of the pattern matching characters is found in the appendix "Pattern Matching."

Table 1-4 Pattern and Character Matching

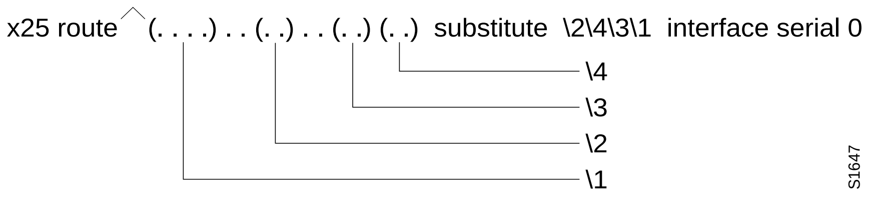
Examples:

This example indicates that X.25 calls to addresses whose first four Data Network Identification Code (DNIC) digits are 1111 should be routed through interface serial 3, but that the DNIC field in the addresses presented to the equipment connected to that interface should be changed to 2222. The \1 characters in the rewrite pattern indicate that the portion of the original address matched by the characters .* should be inserted in the rewritten address.

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

A more contrived example intended to illustrate the power of the rewriting scheme follows:

Figure 1-2 X.121 Address Translation Scheme



x25 route ^(. . . .) . . (. .) . . (. .) (. .) substitute \2\4\3\1 interface serial 0
\4
\3
\2
\1
S1647

This sample would cause all X.25 calls with 14-digit called addresses to be routed through interface serial 0. The incoming DNIC field would be moved to the end of the address. The fifth, sixth, ninth, and tenth digits would be deleted, and the thirteenth and fourteenth would be moved before the eleventh and twelfth.

In the following example, if a call comes in on interface serial 0 and matches any X.121 pattern, the call will be accepted for the given type of connectivity.

```
x25 route .* alias serial 0
```

In the following example, the call will be accepted, because this VAX-to-X.121 address will be treated as a local address as well as the address given in the **x25 address** command for interface serial 0.

```
x25 route ^vax-x121-address serial 0
```

Configuring PVCs on an X.25 Switch

You may configure X.25 Permanent Virtual Circuits (PVCs) in the X.25 switching software. This means that DTEs that require permanent circuits can be connected to the Cisco router acting as an X.25 switch and have a properly functioning connection. X.25 RESETs will be sent indicating when the circuit comes up or goes down.

Use the **x25 pvc** interface subcommand to configure a PVC for a given interface. The syntax of this command follows.

x25 pvc *pvc-number* **interface** *interface-name pvc-number*

The argument *pvc-number* is the PVC number that will be used on the local interface (as defined by the primary interface command). The argument *interface-name* is the interface type and unit number (serial 0, for example), as specified by the **interface** command.

Example:

A PVC is linked across two serial interfaces on the same device. In this type, the alternate 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 as this example illustrates.

```
interface serial 0
encapsulation x25
```

```
x25 pvc 1 interface serial 1 1
interface serial 1
encapsulation x25
x25 pvc 1 interface serial 0 1
```

When you are configuring X.25 to use a PVC, you must ensure that no traffic is sent towards the remote router between the time the **X.25 map** command is issued and the time that **X25 pvc** command is issued. Otherwise, the local router will create an SVC, and then the **pvc** command will not be allowed.

Map entries with the Broadcast attribute are particularly likely to get traffic, due to routing protocol traffic. The simplest way to ensure that no traffic is sent while configuring is to shut down the interface while configuring it for a PVC.

X.25 Switching Configuration Examples

The following examples illustrate how to enable an X.25 switch, how to enable call forwarding, and how to configure a router on a TYMNET/PAD switch to accept and forward calls.

Example 1:

This configuration 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
```

This routing table forwards calls for X.121 address 100 out the interface serial 0. Otherwise, if the X.121 address contains 100 anywhere within it and contains no Call User Data or the Call User Data is not the string pad, it is forwarded onto serial 1. If the X.121 address contains 100 somewhere within and the Call User Data 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 it does match, it is 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 Cisco router at 10.2.0.2 will then route the call according to its X.25 routing table.

Example 2:

This configuration configures a Cisco router that sits on a Tymnet PAD/switch to accept calls and have them forwarded to a DEC VAX system. This feature permits running X.25 network over a generalized, already existing IP network, thereby making it unnecessary to get another physical line for one protocol.

The Cisco 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
```

This would 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 would be forwarded to the *cisco-on-tymnet* address using its IP address. This would take all outgoing calls from the VAX and send them to *cisco-on-tymnet* for further processing.

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

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

This forces all calls with the VAX X.121 address to be sent to the Cisco router with the VAX connected to it. All other calls with X.121 addresses will be forwarded out to Tymnet. If Tymnet can route them, then a CALL ACCEPTED packet will be returned and everything will proceed normally. If Tymnet can not handle it, it will clear the call and the CLEAR REQUEST packet will be forwarded back toward the VAX.

Setting the X.25 Level 3 Parameters

Once you establish X.25 Level 3 encapsulation, you can set X.25 Level 3 parameters. These parameters are described in the following sections.

Note: If you connect a router to an X.25 network, use the parameters set by the network administration. Also, note that the X.25 Level 2 parameters described in “Setting the X.25 Level 2 (LAPB) Parameters” earlier in this chapter affect X.25 Level 3 operations.

Setting the X.25 Interface Address

To set the X.121 address of a particular network interface, use the **x25 address** subcommand. The address is assigned by the X.25 network:

x25 address *X.121-address*

The argument *X.121-address* is a variable-length X.121 address.

Example:

The following subcommand sets the X.121 address of the current interface to address 2.

```
x25 address 2
```

The value must match that assigned by the X.25 network.

The section “DDN X.25 Dynamic Mapping,” earlier in this chapter, describes the addressing scheme for DDN X.25 networks.

Configuring the Virtual Circuit Channel Sequence

An important part of X.25 operation is the virtual circuit channel sequence. This sequence is a range of virtual circuit channel numbers broken into four groups (listed here in numerically increasing order):

1. Permanent virtual circuits
2. Incoming calls
3. Incoming and outgoing (two-way) calls
4. Outgoing calls

Several X.25 parameters determine the numerical ranges of the last three groups; the range for permanent virtual circuits falls numerically below the incoming call range.

X.25 communications devices use the virtual circuit channel sequence when allocating virtual circuits. When initiating a call, these devices must search for an available channel in the sequence in one of two ways. For outgoing calls (made by a DTE device), the search starts at the upper end of the outgoing call range and proceeds in the direction of decreasing channel numbers. The search continues until the device finds an available channel or reaches the lower limit of the two-way call range.

For incoming calls (handled by a DCE device), the search for channel numbers to allocate starts at the lower end of the incoming call range, and proceeds in the direction of increasing channel numbers. The search continues until the device finds an available channel or reaches the upper limit of the two-way call range. The DTE and DCE devices fail to find an available channel only if the overall sequence range is very small or when all of the channels are in use.

To set the upper- and lower-limit parameters of the channel ranges, use the **x25** subcommand keywords listed in Table 1-5. Each keyword takes a channel number as its argument. Note that the values for these parameters must be the same on both ends of an X.25 link.

The default lower limit for all channel ranges on the router is 1; the default upper limit for all ranges is 1024. These defaults reflect a simple approach to allocating the channel ranges: assign the same low value to all lower limits, and assign the same high value to all upper limits. This approach causes the three ranges to overlap and become one large range.

Table 1-5 Range Limit Keywords for the Virtual Circuit Channel Sequence

Keyword	Limit Type	Range	Default
lic	Lower limit, incoming call range	1-4095	1
hic	Upper limit, incoming call range	1-4095	1024
ltc	Lower limit, two-way call range	1-4095	1
htc	Upper limit, two-way call range	1-4095	1024
loc	Lower limit, outgoing call range	1-4095	1
hoc	Upper limit, outgoing call range	1-4095	1024

S1614

Setting the Highest Incoming Channel

The **hic** keyword sets the highest incoming channel (HIC).

x25 hic *channel*

The argument *channel* is a channel number from 1 through 4095. The default value is 1024.

Setting the Highest Outgoing Channel

The **hoc** keyword sets the highest outgoing channel (HOC).

x25 hoc *channel*

The argument *channel* is a channel number from 1 through 4095. The default value is 1024.

Setting the Highest Two-Way Channel

The **htc** keyword sets the highest two-way channel (HTC).

x25 htc *channel*

The argument *channel* is a channel number from 1 through 4095. The default value is 1024.

Setting the Lowest Incoming Channel

The **lic** keyword sets the lowest incoming channel (LIC).

x25 lic *channel*

The argument *channel* is a channel number from 1 through 4095. The default value is one.

Setting the Lowest Outgoing Channel

The **loc** keyword sets the lowest outgoing channel (LOC).

x25 loc *channel*

The argument *channel* is a channel number from 1 through 4095. The default value is one.

Setting the Lowest Two-Way Channel

The **ltc** keyword sets the lowest two-way channel (LTC).

x25 ltc *channel*

The argument *channel* is a channel number from 1 through 4095. The default value is one.

Example:

The following commands set these channels.

```
01-20: — Incoming
01-20: — Either incoming or outgoing
01-20: — Outgoing
!
x25 hic 20
x25 htc 20
x25 hoc 20
```

Maintaining Virtual Circuits

The router can clear a switched virtual circuit (SVC) after a set period of inactivity. To set this period, use the **x25 idle** interface subcommands:

x25 idle *minutes*
no x25 idle

The argument *minutes* is the number of minutes in the period. The default value is zero, which causes the router to keep the SVC open indefinitely. Both calls originated and terminated by the router are cleared. The **no x25 idle** command returns the default.

To increase throughput across networks, you can establish up to eight SVCs to a host. To specify the maximum number of SVCs that can be open simultaneously to one host, use the **x25 nvc** interface subcommand:

x25 nvc *count*

The argument *count* is a circuit count from 1 to 8; the default is 1.

Note: This affects the default value for the number of SVCs. It does not affect the NVC value for any **x25 map** commands that have already been configured.

Configuring the Ignore VC Timer

Upon receiving a Clear Request for an outstanding Call Request, the X.25 support code immediately tries another Call Request, if it has more traffic to send. This can overrun some X.25 switches. To prevent this problem, use the **x25 hold-vc-timer** configuration commands:

x25 hold-vc-timer *minutes*
no x25 hold-vc-timer

The argument *minutes* is the number of minutes to prevent calls to a previously failed destination. Incoming calls will still be accepted. The default value is 0, and the **no x25 hold-vc-timer** command restores this default.

Configuring the X.25 Level 3 Retransmission Timers

The X.25 Level 3 retransmission timers determine how long the router must wait before retransmitting various Request packets. You can set these timers independently using the **x25** subcommand keywords listed in Table 1-6. Each keyword requires a time value in seconds as its argument. The last column shows the default timer values, in seconds. Four of the timers apply to DTE devices, and the other four apply to DCE devices.

Table 1-6 Retransmission Timer Keywords and Defaults

Keyword (DTE/DCE)	Affected Request Packet	Time(seconds) (DTE/DCE)
t20/t10	Restart Request	180/60
t21/t11	Call Request	200/180
122/t12	Reset Request	180/60
t23/t13	Clear Request	180/60

S1616

Setting the DTE Restart Request Retransmission Timer

The **t20** keyword sets the limit for the Restart Request retransmission timer (T20) on DTE devices.

x25 t20 *seconds*

The argument *seconds* is a time value in seconds. The default is 180 seconds.

Setting the DCE Restart Request Retransmission Timer

The **t10** keyword sets the limit for the Restart Request retransmission timer (T10) on DCE devices.

x25 t10 *seconds*

The argument *seconds* is a time value in seconds. The default value is 60 seconds.

Setting the DTE Call Request Retransmission Timer

The **t21** keyword sets the limit for the Call Request retransmission timer (T21) on DTE devices.

x25 t21 *seconds*

The argument *seconds* is a time value in seconds. The default value is 200 seconds.

Setting the DCE Call Request Retransmission Timer

The **t11** keyword sets the limit for the Call Request retransmission timer (T11) on DCE devices.

x25 t11 *seconds*

The argument *seconds* is a time value in seconds. The default value is 180 seconds.

Setting the DTE Reset Request Retransmission Timer

The **t22** keyword sets the limit for the Reset Request retransmission timer (T22) on DTE devices.

x25 t22 *seconds*

The argument *seconds* is a time value in seconds. The default value is 180 seconds.

Setting the DCE Reset Request Retransmission Timer

The **t12** keyword sets the limit for the Reset Request retransmission timer (T12) on DCE devices.

x25 t12 *seconds*

The argument *seconds* is a time value in seconds. The default value is 60 seconds.

Setting the DTE Clear Request Retransmission Timer

The **t23** keyword sets the limit for the Clear Request retransmission timer (T23) on DTE devices.

x25 t23 *seconds*

The argument *seconds* is a time value in seconds. The default value is 180 seconds.

Setting the DCE Clear Request Retransmission Timer

The **t13** keyword sets the limit for the Clear Request retransmission timer (T13) on DCE devices.

x25 t13 *seconds*

The argument *seconds* is a time value in seconds. The default value is 60 seconds.

Updating the X.121 Address

Some X.25 calls, when forwarded by the X.25 switching support, need the calling (source) X.121 address updated to that of the outgoing interface. This is necessary when forwarding calls from private data networks to public data networks.

Outgoing calls forwarded over a specific interface can have their calling X.121 address updated by using the **x25 use-source-address** subcommand. The full syntax is:

x25 use-source-address
no x25 use-source-address

The **no x25 use-source-address** command prevents updating of the source address of outgoing calls.

Setting X.25 Packet Sizes

X.25 networks use maximum input and output packet sizes set by the network administration. You can set the router input and output packet sizes to match those of the network with the **x25** subcommand keywords **ips** and **ops,** respectively:

x25 ips *bytes*
x25 ops *bytes*

The argument *bytes* is a byte count in the range of 128 through 2048. The default value is 128 bytes.

Larger packet sizes are better, because smaller packets require more overhead processing.

Note: Set the **x25 ips** and **x25 ops** keywords to the same value unless your network supports asymmetry between input and output packets.

To send a packet larger than the X.25 packet size over an X.25 virtual circuit, a router 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 with the M-bit set and reassembles them.

Setting the Flow Control Modulus

X.25 supports flow control with a sliding window sequence count. The window counter restarts at zero upon reaching the upper limit, which is called the window modulus. To set the window modulus, use the **x25 modulo** interface subcommand:

x25 modulo *modulus*

The argument *modulus* is either 8 or 128. The default value is eight. The value of the modulo parameter must agree with that of the device on the other end of the X.25 link.

Configuring Packet Acknowledgment

To specify upper limits on the number of outstanding unacknowledged packets, use the commands **x25 win** (for input window) and **x25 wout** (for output window):

x25 win *packets*
x25 wout *packets*

The argument *packets* is a packet count. The packet count for **win** and **wout** can range from one to the window modulus. The default value is two packets.

The **x25 win** command determines how many packets the router can receive before sending an X.25 acknowledgment. The **wout** limit determines how many sent packets can remain unacknowledged before the router uses a hold queue. To maintain high bandwidth utilization, assign these limits the largest number that the network allows.

Note: Set **win** and **wout** to the same value unless your network supports asymmetry between input and output window sizes.

You can instruct the router to send acknowledgment packets when it is not busy sending other packets, even if the number of input packets has not reached the **win** count. This approach improves line responsiveness at the expense of bandwidth. To enable this option, use the **x25 th** subcommand:

x25 th *delay*

The argument *delay* must be between zero and the input window size. A value of one sends one Receiver Ready acknowledgment per packet at all times. The default value is zero, which disables the delayed acknowledgment strategy.

The router sends acknowledgment packets when the number of input packets reaches the count you specify, providing there are no other packets to send. For example, if you specify a count of one, the router can send an acknowledgment per input packet.

Suppressing the Calling Address

To omit the calling address in outgoing calls, use the **x25 suppress-calling-address** interface subcommands:

x25 suppress-calling-address
no x25 suppress-calling-address

The **suppress-calling-address** keyword omits the calling (source) X.121 address in Call Request packets. This option is required for networks that expect only subaddresses in the calling address field. The calling address is sent by default.

Use the **no x25 suppress-calling-address** subcommand to reset this subcommand to the default state.

Suppressing the Called Address

To omit the called address in outgoing calls, use the **x25 suppress-called-address** interface subcommands:

x25 suppress-called-address
no x25 suppress-called-address

The **suppress-called-address** keyword omits the called (destination) X.121 address in Call Request packets. This option is required for networks that expect only subaddresses in the called address field. The called address is sent by default.

Use the **no x25 suppress-called-address** subcommand to reset this subcommand to the default state.

Defining a Packet Hold Queue

To define the number of packets to be held until a virtual circuit is established, use the **x25 hold-queue** interface subcommand:

x25 hold-queue *queue-size*
no x25 hold-queue [*queue-size*]

The argument *queue-size* defines the number of packets. By default, this number is zero. Use the **no x25 hold-queue** command without an argument to remove this command from the configuration file; enter the command with a queue size value of zero to return the default.

Accepting Reverse Charge Calls

To instruct the router to accept all reverse charge calls, use the **x25 accept-reverse** interface subcommands:

x25 accept-reverse
no x25 accept-reverse

The **accept-reverse** keyword causes the interface to accept reverse charge calls by default. This behavior can also be configured on a per-peer basis using the **x25 map** subcommand.

The **no x25 accept-reverse** command disables this facility.

Forcing Packet-Level Restarts

To force a packet-level restart when the link level is restarted, use the **x25 linkrestart** interface subcommands:

x25 linkrestart
no x25 linkrestart

This command restarts X.25 Level 2 (LAPB) when errors occur. This behavior is the default and is necessary for networks that expect this behavior. Use the **no x25 linkrestart** to turn this feature off.

Setting the Packet Network Carrier

To set the packet network carrier, use the **x25 rpoa** interface subcommands:

x25 rpoa *name number*
no x25 rpoa *name*

The **x25 rpoa** interface subcommand specifies a list of transit RPOAs to use, referenced by name. The argument *name* must be unique with respect to all other RPOA names. It is used in the **x25 facility** and **x25 map** interface subcommands. The argument *number* is a number that is used to describe an RPOA. The **no x25 rpoa** command removes the specified name.

Setting X.25 Parameters on a Per-Call Basis

To override interface settings on a per-call basis, use the **x25 facility** interface subcommand. The full syntax of the command follows:

x25 facility *keyword argument*
no x25 facility *keyword argument*

The command enables X.25 facilities, which are sent between DTE and DCE devices to negotiate certain link parameters.

The argument *keyword* specifies one of the following keywords, followed by the required *argument*:

- **cug** *number*—Specifies a Closed User Group *number* from 1 through 99 to provide an extra measure of network security.
- **packetsize** *in-size out-size*—Sets the size in bytes of input packets *(in-size)* and output packets *(out-size)*. Both values should be the same.
- **reverse**—Reverses charges on all Call Request packets from the interface.
- **windowsize** *in-size out-size*—Sets the packet count for input windows (*in-size*) and output windows (*out-size*). Both values should be the same.
- **throughput** *in out*—Sets the requested throughput values for input and output throughput across the network.
- **rpoa** *name*—Specifies the list of transit Recognized Private Operating Agencies (RPOAs) to use in outgoing Call Request packets.

The **no x25 facility** command with the appropriate keyword and argument removes the facility.

Maintaining X.25

To clear all virtual circuits at once, use the privileged EXEC command **clear x25-vc.** This command takes an interface type keyword (usually **serial**) and a unit number as arguments to identify the interface with which the virtual circuits are associated.

To clear a particular virtual circuit, add the two arguments described above the **clear x25-vc** command, and include a logical circuit number (LCN) value as a third argument. The command syntax is:

clear x25-vc *interface unit* [*lcn*]

The **clear x25-vc** command clears all X.25 virtual circuits at once. The argument *interface* is the interface type. The argument *unit* is the interface unit number. The optional argument *lcn* clears the specified virtual circuit.

Monitoring X.25 Level 3 Operations

The router provides EXEC **show** commands to provide information on interface operation and virtual circuit operation. Use the EXEC command **show interfaces** to display interface parameters and statistics. Use the EXEC command **show x25 vc** to display virtual circuit parameters and statistics.

Displaying Interface Parameters and Statistics

To display parameter information for an interface using X.25 Level 3 protocol, use the EXEC command **show interfaces.** For X.25 interfaces, the following is an example of output:

```
X25 address 000000010100, state R1, modulo 8, idle 0, timer 0, nvc 1
Window size: input 2, output 2, Packet size: input 128, output 128
Timers: T20 180 T21 200 T22 180 T23 180 TH 0
Channels: Incoming 1-1024 Two-way 1-1024 Outgoing 1-1024
RESTARTs 1/20 CALLs 1000+2/1294+190/0+0 DIAGs 0/0
```

On the first line, the `address` field indicates the calling address used in the Call Request packet. The `state` field indicates the state of the interface: `R1` is the normal ready state, `R2` indicates the DTE not-ready state, and `R3` indicates the DCE not-ready state. If the state is `R2` or `R3`, the device is awaiting acknowledgment for a Restart Request packet. The `modulo` field displays the modulo value, which determines the sequence numbering scheme used. The `idle` field shows the number of minutes the router waits before closing idle virtual circuits. The `timer` field displays the value of the interface timer, which is zero unless the interface state is `R2` or `R3`. The `nvc` field displays the maximum number of simultaneous virtual circuits permitted to and from a single host.

On the second line, the `Window size` and `Packet size` fields show the default window and packet sizes for the interface. Each virtual circuit can override these values using facilities specified with the **x25 map** or **x25 facility** subcommands.

The third line shows the values of the Request packet timers (T10 through T13 for a DCE device, and T20 through T23 for a DTE device). The fourth line shows the channel sequence ranges. The last line shows packet statistics for the interface using these formats:

```
sent/received\ successful+failed\ callssent+callssentfailed/\
callsreceived+callsreceivedfailed/\
callsforwarded+callsforwardedfailed\
```

Displaying Virtual Circuit Parameters and Statistics

The EXEC command **show x25 vc** displays the details of the active X.25 switched virtual circuits. To examine a particular virtual circuit, add an LCN argument to the **show x25 vc** command. The following is example output:

```
LC1: 1, State: D1, Interface: Serial0
Started 0:55:03, last input 0:54:56, output 0:54:56
Connected to IP [10.4.0.32] <–>000000320400 Precedent: 0
Window size input: 7, output: 7
Packet size input: 1024, output: 1024
PS: 2 PR: 6 Remote PR: 2 RCNT: 1 RNR: FALSE
```

```
Retransmits: 0   Timer (secs): 0  Reassembly (bytes): 0
Held Fragments/Packets: 0/0
Bytes 1111/588 Packets 18/22 Resets 0/0 RNRs 0/0 REJs 0/0 INTs 0/0
```

On the first line, the `LCI` field displays the virtual circuit number. The `State` field displays the state of the virtual circuit (which is independent of the states of other virtual circuits); D1 is the normal ready state. (See the CCITT X.25 recommendation for a description of virtual circuit states.) The `Interface` field shows the interface used for the virtual circuit.

On the second line, the `Started` field shows the time elapsed since the virtual circuit was created, the `last input` field shows time of last input, and the `output` field shows time of last output.

On the third line, the `Connected to` field shows in brackets the network-protocol address and then the X.121 address. The `Precedent` field, which appears only if you have specified DDN encapsulation, indicates IP precedence.

The fourth and fifth lines show window and packet sizes. These sizes can differ from the interface default values if facilities were offered and accepted in the Call Request and Call Accepted packets.

On the sixth line, the `PS` and `PR` fields show the current send and receive sequence numbers, respectively. The `Remote PR` field shows the last `PR` number received from the other end of the circuit. The `RCNT` field shows the count of unacknowledged input packets. The `RNR` field shows the state of the `Receiver Not Ready` flag; this field is true of the network sends a receiver-not-ready packet.

On the seventh line, the `Retransmits` field shows the number of times a packet has been retransmitted. The `Timer` field shows a nonzero time value if a packet has not been acknowledged or if virtual circuits are being timed for inactivity. The `Reassembly` field shows the number of bytes received for a partial packet (a packet in which the more data bit is set).

On the eighth line, the fragments part of the `Held Fragments/Packets` field shows the number of X.25 packets being held. (In this case, `Fragments` refers to the X.25 fragmentation of higher-level data packets.) The Packets part of the `Held Fragments/Packets` field shows the number of higher-level Protocol packets currently being held pending the availability of resources, such as the establishment of the virtual circuit.

On the last line, the `Bytes` field shows the total numbers of bytes sent and received. The `Packets, Resets, RNRs, REJs,` and `INTs` fields show the total sent and received packet counts of the indicated types. (RNR is Receiver Not Ready, REJ is Reject, and INT is Interrupt).

Debugging X.25

When installing an X.25 link, you can use the EXEC commands **debug** with different keywords as follows:

debug x25

This command enables the monitoring of all X.25 traffic.

debug x25-events

This command enables the monitoring of all X.25 traffic but does not display information about X.25 data or acknowledgment packets.

debug x25-vc *number*

This command allows you to watch the specified port *number* with a virtual circuit using the **debug x25** and **debug x25-events** commands.

The following is example output:

```
Serial0: X25 I R1 RESTART (5) 8 lci 0 cause 7 diag 250
Serial0:  X25 0 R1 RESTART CONFIRMATION (3) 8 lci 0
Serial0: X25 0 P2 CALL REQUEST (19) 8 lci 1
From(14): 31250000000101 To(14): 31109090096101
Facilities (0)
Serial0: X25 0 P6 CLEAR REQUEST (5) 8 lci 1 cause diag 122
```

For each event, the first field identifies the interface on which the activity occurred, and the second field indicates that it was an X.25 event. The third field indicates whether the X.25 packet was input (`I`) or output (`O`). The fourth field is the state of the interface: `R1` is the normal ready state, `R2` indicates the DTE not-ready state, and `R3` indicates the DCE not-ready state.

The fifth field is the type of the X.25 packet that triggered the event, and the sixth field (in parentheses) gives the total length of the X.25 packet in bytes. The seventh field is the window modulus. The eighth field (labeled `lci`) shows the virtual circuit number. The ninth field (labeled `cause`) gives the cause code, and the tenth field (labeled `diag`) gives the diagnostic code.

For Call Request and Call Connected packets, the router shows additional information on separate lines. The `From` field shows the calling X.121 address and the `To` field shows the called X.121 address. The number in parentheses after the field name—`(14)` for example—specifies the number of digits in the address. The `Facilities` field indicates the length (in bytes) of the requested facilities and the facilities contents.

X.25 Global Configuration Command Summary

This section provides an alphabetically arranged summary of the X.25 global configuration commands.

[**no**] **x25 route** [*# position*]*x121-pattern* [**cud** *pattern*] **interface** *interface-name*
[**no**] **x25 route** [*# position*]*x121-pattern* [**cud** *pattern*] **ip** *ip-address*
[**no**] **x25 route** [*# position*]*x121-pattern* [**cud** *pattern*] **alias** *interface-name*
[**no**] **x25 route** [*#position*]*x121-pattern* [**substitute-source** *rewrite-pattern*]
[**substitute-dest** *rewrite-pattern*] [**cud** *pattern*] *interface destination*

Inserts or removes an entry in the X.25 routing table. The *x121-pattern pattern* parameter is the X.121 address of the called destination and is required. The **alias** keyword permits a way for other X.121 addresses to be treated as local. The **substitute-source** keyword allows substitution of the calling address. The argument *rewrite-pattern* replaces the called or calling X.121 address in routed X.25 packets.

[**no**] **x25 routing**

Enables or disables X.25 switching. X.25 calls will not be forwarded until this command is issued. The command **no x25 routing** disables the forwarding of X.25 calls.

X.25 Interface Subcommand Summary

This section provides an alphabetical list of all the interface subcommands used in the X.25 interface.

[**no**] **x25 accept-reverse**

Instructs the router to accept all reverse charge calls. This behavior can also be configured on a per-peer basis using the **x25 map** subcommand. The **no** form of the command resets the default state.

x25 address *X.121-address*

Sets the X.121 address of a particular network interface. The address is assigned by the X.25 network. The argument *X.121-address* is a variable-length X.121 address.

[**no**] **x25 default** *protocol*

Specifies or removes the protocol assumed by the CPT to interpret calls with unknown Call User Data. The argument *protocol* sets the default protocol and is either **ip** or **pad**.

[**no**] **x25 facility** *keyword argument*

Enables and disables the X.25 facilities, which are sent between DTE and DCE devices to negotiate certain link parameters. The argument *keyword* specifies one of the following keywords, followed by the required *argument*:

- **cug** *number*—Specifies a Closed User Group *number* from 1 through 99 to provide an extra measure of network security.
- **packetsize** *in-size out-size*—Sets the size in bytes of input packets *(in-size)* and output packets *(out-size)*. Both values should be the same.
- **reverse**—Reverses charges on all Call Request packets from the interface.
- **windowsize** *in-size out-size*—Sets the packet count for input windows (*in-size*) and output windows (*out-size*). Both values should be the same.
- **throughput** *in out*—Sets the requested throughput values for input and output throughput across the network.
- **rpoa** *name*—Specifies the list of transit RPOAs to use in outgoing Call Request

packets.

x25 hic *channel*

Sets the highest incoming channel (HIC). The argument *channel* is a channel number from 1 through 4095. The default value is 1024.

x25 hoc *channel*

Sets the highest outgoing channel (HOC). The argument *channel* is a channel number from 1 through 4095. The default value is 1024.

[**no**] **x25 hold-vc-timer** *minutes*

Prevents overruns on X.25 switches for traffic through the VCs. The argument *minutes* is the number of minutes to prevent calls to a previously failed destination. Incoming calls will still be accepted.

x25 htc *channel*

Sets the highest two-way channel (HTC). The argument *channel* is a channel number from 1 through 4095. The default value is 1024.

[**no**] **x25 idle** *minutes*

Sets the period of inactivity once an SVC has been cleared. The argument *minutes* is the number of minutes in the period. Both calls originated and terminated by the router are cleared. The default value is 0 (zero), which causes the router to keep the SVC open indefinitely. The **no** variation restores this default.

[**no**] **x25 ip-precedence**

Enables or disables the ability to open a new virtual circuit based on the IP Type of Service (TOS) field. By default, Cisco routers open one virtual circuit for each type of service.

x25 ips *bytes*
x25 ops *bytes*

Set the router packet size to match those of the network. The **ips** keyword specifies the router input packet size while the keyword **ops** specifies the router output packet size. The argument *bytes* is a byte count in the range of 128 through 1024. The default value is 128 bytes. Larger packet sizes are better, because smaller packets require more overhead processing.

Note: Set the **x25 ips** and **x25 ops** keywords to the same value unless your network supports asymmetry between input and output packets.

x25 lic *channel*

Sets the lowest incoming channel (LIC). The argument *channel* is a channel number from 1 through 4095. The default value is one.

[**no**] **x25 linkrestart**

Forces a packet-level restart when the link level is restarted and restarts X.25 Level 2 (LAPB) when errors occur. This behavior is the default and is necessary for networks that expect this behavior. The **no** form of the command resets the default state.

x25 loc *channel*

Sets the lowest outgoing channel (LOC). The argument *channel* is a channel number from 1 through 4095. The default value is one.

x25 ltc *channel*

Sets the lowest two-way channel (LTC). The argument *channel* is a channel number from 1 through 4095. The default value is one.

[**no**] **x25 map** *protocol-keyword protocol-address X.121-address* [*option1 … option*]

Specifies a network-protocol-to-X.121 address mapping such as Internet-to-X.121 or DECnet-to-X.121. The argument *protocol-keyword* can be one of these protocol types: **ip**, **decnet**, **chaos**, **xns** , **novell**, **appletalk**, **vines**, **apollo**, **pup**. The *address* arguments specify the network-protocol-to-X.121 mapping. The *option* arguments add certain features to the mapping specified, and can be any of the following, up to six. They must be specified in the order listed.

- **reverse**—Specifies reverse charging for outgoing calls.
- **accept-reverse**—Causes the router to accept incoming reverse-charged calls. If this option is not present, the router clears reverse charge calls.
- **broadcast**—Causes the router to direct any broadcasts sent through this interface to the specified X.121 address. This option is needed when dynamic routing protocols are being used to access the X.25 network.
- **cug** *number*—Specifies a Closed User Group number (from 1 to 99) for the mapping in the outgoing call.
- **nvc** *count*—Sets the number of virtual circuits (VCs) for this map/host. The default *count* is the **x25 nvc** setting of the interface. A maximum number of eight VCs can

be configured for a single map/host.

- **packetsize** *in-size out-size*—Specifies input packet size *in-size* and output packet size *out-size* for the mapping in the outgoing call.
- **windowsize** *in-size out-size*—Specifies input window size *in-size* and output window size *out-size* for the mapping in the outgoing call.
- **throughput** *in out*—Requests the amount of bandwidth through the X.25 network.
- **modulo** *size*—Specifies the maximum window size for this map. The argument *size* permits windows of 8 or 128 on the same interface.
- **nuid** *username password*—Specifies that a network ID facility be sent in the outgoing call with the specified user name and password.

x25 modulo *modulus*

Sets the modules. The argument *modulus* is either 8 or 128. The default value is eight. The value of the modulo parameter must agree with that of the device on the other end of the X.25 link.

x25 nvc *count*

Specifies the maximum number of switched virtual circuits that can be open simultaneously to one host. The argument *count* is a circuit count from 1 to 8; the default is 1.

[**no**] **x25 pvc** *circuit protocol-keyword protocol-address*

Establishes or deletes Permanent Virtual Circuits (PVCs). The argument *circuit* is a virtual circuit channel number and it must be less than the lower limit of the incoming call range in the virtual circuit channel sequence (set using the **lic** keyword). The argument *protocol-keyword* can be one of the protocol types mentioned above. The argument *protocol-address* is that of the host at the other end of the PVC.

Note: You must specify the required network-protocol-to-X.121 address mapping with an **x25 map** subcommand before you can set up a PVC.

---

x25 pvc *pvc-number* **interface** *interface-name pvc-number*

Configures a PVC for a given interface. The argument *pvc-number* is the PVC number that will be used on the local interface (as defined by the primary interface command). The argument *interface-name* is the interface type and unit number (serial 0, for example), as specified by the **interface** keywords.

[**no**] **x25 rpoa** *name number*

Specifies a list of transit RPOAs to use, referenced by name. The argument *name* must be unique with respect to all other RPOA names. It is used in the **x25 facility** and **x25 map** interface subcommands. The argument *number* is a number that is used to describe an RPOA.

[**no**] **x25 suppress-calling-address**

Omits the calling (source) X.121 address in Call Request packets. This option is required for networks that expect only subaddresses in the calling address field. The calling address is sent by default. The **no** form of the command resets the default state.

x25 t10 *seconds*

Sets the limit for the Restart Request retransmission timer (T10) on DCE devices. The argument *seconds* is a time value in seconds. The default value is 60 seconds.

x25 t11 *seconds*

Sets the limit for the Call Request retransmission timer (T11) on DCE devices. The argument *seconds* is a time value in seconds. The default value is 180 seconds.

x25 t12 *seconds*

Sets the limit for the Reset Request retransmission timer (T12) on DCE devices. The argument *seconds* is a time value in seconds. The default value is 60 seconds.

x25 t13 *seconds*

Sets the limit for the Clear Request retransmission timer (T13) on DCE devices. The argument *seconds* is a time value in seconds. The default value is 60 seconds.

x25 t20 *seconds*

Sets the limit for the Restart Request retransmission timer (T20) on DTE devices. The argument *seconds* is a time value in seconds. The default is 180 seconds.

x25 t21 *seconds*

Sets the limit for the Call Request retransmission timer (T21) on DTE devices. The argument *seconds* is a time value in seconds. The default value is 200 seconds.

x25 t22 *seconds*

Sets the limit for the Reset Request retransmission timer (T22) on DTE devices. The argument *seconds* is a time value in seconds. The default value is 180 seconds.

x25 t23 *seconds*

Sets the limit for the Clear Request retransmission timer (T23) on DTE devices. The argument *seconds* is a time value in seconds. The default value is 180 seconds.

x25 th *delay*

Instructs the router to send acknowledgment packets when it is not busy sending other packets, even if the number of input packets has not reached the **win** count, which improves line responsiveness at the expense of bandwidth. The argument *delay* must be between zero and the input window size. A value of one sends one Receiver Ready acknowledgment per packet at all times. The default value is zero, which disables the delayed acknowledgment strategy.

[**no**] **x25 use-source-address**

Updates the source address of outgoing calls forwarded over a specific interface use the following command. The **no** variation prevents the update.

x25 win *packets*
x25 wout *packets*

Set the upper limits on the number of outstanding unacknowledged packets. The **win** keyword specifies the upper limits of the number of outstanding unacknowledged packets in the input window., and determines how many packets the router can receive before sending an X.25 acknowledgment.

The **wout** keyword specifies the upper limits of the number of outstanding unacknowledged packets in the output window. The **wout** limit determines how many sent packets can remain unacknowledged before the router uses a hold queue.

To maintain high bandwidth utilization, assign these limits the largest number that the network allows. The argument *packets* is a packets count. The packet count for **win** and **wout** can range from one to the window modulus. The default value is two packets.

Note: Set **win** and **wout** to the same value unless your network supports asymmetry between input and output window sizes.

Configuring Frame Relay

This section describes frame relay configuration, Cisco Systems' implementation of frame relay, the protocols supported, and the hardware needed to operate with a frame relay network.

Frame relay is described as an encapsulation method and is directed mainly to users with large T1 network installations. Cisco's implementation meets the *Frame Relay Interface* specification produced by Northern Telecom, Digital Equipment Corporation, StrataCom, and Cisco Systems. Cisco's implementation also conforms to the Link Access Procedure (LAP-D) defined by the CCITT under its I-series (ISDN) recommendation as I122, "Framework for Additional Packet Model Bearer Services." Cisco's frame relay implementation currently supports routing on IP, DECnet, AppleTalk, Novell IPX, and VINES, and transparent bridging.

The Cisco network server supports the local management interface (LMI), as specified in the joint *Frame Relay Interface* specification. The LMI includes support for a keepalive mechanism, a multicast group, and a status message.

The keepalive mechanism provides an exchange of information between the network server and the switch to verify data is flowing. The multicast mechanism provides the network server with its local data link connection identifier (DLCI) and the multicast DLCI. The status mechanism provides an on-going status report on the DLCIs known by the switch.

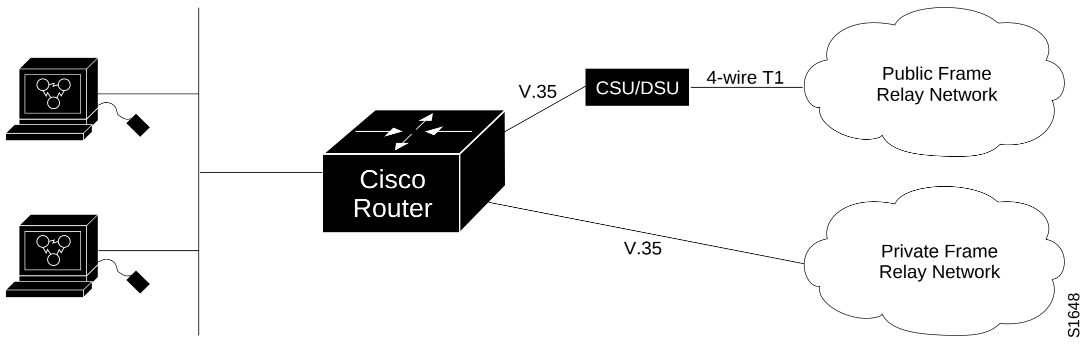
Configuring the Hardware

The following hardware configuration is required for frame relay connections:

- Each router connects directly to the frame relay switch.
- Each router connects directly to a CSU/DSU (Channel Service Unit/Digital Service Unit) first and the CSU/DSU is connected to a remote frame relay switch.

The CSU/DSU converts V.35 or RS-449 signals to the properly coded T1 transmission signal for successful reception by the frame relay network. Figure 1-3 illustrates the connections between the different components.

Figure 1-3 Frame Relay Physical Connection



The frame relay interface actually consists of one physical connection between the network server and the switch that provides the service. This single physical connection provides direct connectivity to each device on a network, such as a Stratacom FastPacket wide area network, using only a single connection.

Specifying Frame Relay Encapsulation

Use the **encapsulation frame-relay** interface subcommand to specify frame relay encapsulation on a specific interface.

encapsulation frame-relay

Example:

These commands configure frame relay encapsulation on interface serial 1.

```
interface serial 1
encapsulation frame-relay
```

Setting the Frame Relay Keepalive Timer

The **frame-relay keepalive** interface subcommand enables and disables the LMI mechanism for serial lines using the frame relay encapsulation. The full syntax of this command follows.

frame-relay keepalive *number*
no frame-relay keepalive

The argument *number* defines the keepalive interval. The interval must be set, and must match the interval set on the switch. The default keepalive interval is ten seconds.

Note: The **frame-relay keepalive** and **keepalive** commands perform the same function; both commands enable the keepalive sequence. The keepalive sequence is part of the LMI protocol, so these commands also control the enabling and disabling of the LMI.

Example:

This command sets the keepalive timer on the Cisco router for a period that is two or three seconds faster (shorter interval) than the interval set on the keepalive timer of the frame relay switch. The difference in keepalive intervals assures proper synchronization between the Cisco router and the frame relay switch.

```
frame-relay keepalive 8
```

Mapping Between an Address and the DLCI

The **frame-relay map** subcommand defines the mapping between an address and the

DLCI used to connect to the address. There can be many DLCIs known by a network server that can send data to many different places, but all this data will be multiplexed over the one physical link. The frame relay map tells the network server how to get from a specific protocol and address pair to the correct DLCI. The full syntax of this command follows.

frame-relay map *protocol protocol-address DLCI* [**broadcast**]
no frame-relay map *protocol protocol-address*

The argument *protocol* is one of these keywords: **ip**, **decnet**, **appletalk**, **xns**, **novell**, **vines**, or **clns**. The keyword is followed by the corresponding protocol address and the DLCI number.

This variation of the **frame-relay map** command is used for the ISO CLNS protocol:

frame-relay map clns *DLCI* **broadcast**

This variation of the **frame-relay map** command is used for bridging:

frame-relay map bridge *DLCI* **broadcast**

In both these variations, there is no need to specify a protocol address.

The optional keyword **broadcast** specifies that broadcasts should be forwarded to this address when the multicast is not enabled. The default is not to forward broadcasts. The **broadcast** keyword is required for ISO CLNS and bridging applications.

The **no frame-relay map** subcommand with the appropriate arguments deletes the entry.

Examples of command use follow; see the chapter "Configuring Transparent Bridging" for the procedures to configure bridging on frame relay.

Example 1:

This command maps IP address *131.108.123.1* to *DLCI 100*. Broadcasts are not forwarded.

```
frame-relay map IP 131.108.123.1 100
```

Example 2:

This command uses *DLCI 144* for bridging.

```
frame-relay map bridge 144 broadcast
```

Example 3:

This command uses *DLCI 125* for ISO CLNS routing.

```
frame-relay map clns 125 broadcast
```

Requesting Short Status Messages

The **frame-relay short-status** interface subcommand instructs the network server to request the short status message from the switch (see version 2.3 of the joint *Frame Relay Interface* specification). The full syntax of this command follows:

frame-relay short-status
no frame-relay short-status

The default is to request the full status message. Use the **no frame-relay short-status** command to override the default.

Example:

This command returns the interface to the default state of requesting full status messages.

```
no frame-relay short-status
```

Setting a Local DLCI

The **frame-relay local-dlci** interface subcommand sets the source DLCI for use when the LMI is not supported. If LMI is supported and the multicast information element is present, the network server sets its local DLCI based on information provided via the LMI. The full syntax of this command follows:

frame-relay local-dlci *number*
no frame-relay local-dlci

The argument *number* is the local, or source, DLCI number. The **no frame-relay local-dlci** command removes DLCI number.

Note: The **frame-relay local-dlci** command is provided mainly to allow testing of the frame relay encapsulation in a setting where two routers are connected back to back. This command is not required in a live frame relay network.

Example:

This command specifies 100 as the local DLCI.

```
frame-relay local-dlci 100
```

Defining a DLCI for Multicast

The **frame-relay multicast-dlci** interface subcommand define or remove a DLCI to be used for multicasts. The full syntax of this command follows.

frame-relay multicast-dlci *number*
no frame-relay multicast-dlci

The router sends broadcast packets to all DLCIs that have the **broadcast** flag set in the map entry. Network transmissions (packets) sent to a multicast DLCI are delivered to all network servers defined as members of the multicast group. The multicast DLCI is identified by the argument *number*. (Note that this is *not* the multicast group number, which is an entirely different value.) The **no frame-relay multicast-dlci** command removes the multicast group.

Note: The **frame-relay multicast-dlci** command is provided mainly to allow testing of the frame relay encapsulation in a setting where two routers are connected back to back. This command is not required in a live frame relay network.

Example:

This command specifies 1022 as the multicast DLCI.

```
frame-relay multicast-dlci 1022
```

Frame Relay Configuration Examples

The following examples are included to show you how you may configure your router to support frame relay connections.

Two Routers in Static Mode

The following examples illustrate how to configure two routers for static mode.

Example 1—First Router:

```
interface  serial 0
ip address  131.108.64.2 255.255.255.0
encapsulation  frame-relay
frame-relay keepalive 10
frame-relay map ip  131.108.64.1 43
```

Example 2—Second Router:

```
interface  serial 0
ip address  131.108.64.1 255.255.255.0
encapsulation  frame-relay
frame-relay keepalive 10
frame-relay map ip  131.108.64.2 44
```

Routing DECnet Packets

The following example illustrates how to send all DECnet packets destined for address 56.4 out on DLCI 101. In addition, any DECnet broadcasts for interface serial 1 will also be sent on the DLCI.

Example:

```
interface  serial 1
decnet routing 32.6
```

```
encapsulation  frame-relay
frame-relay map decnet 56.4 101 broadcast
```

Routing Novell Packets

The following example illustrates how to send packets destined for Novell address *200.0000.0c00.7b21* out on DLCI 102.

Example:

```
interface  ethernet 0
novell network 2abc
!
interface serial 0
novell network 200
encapsulation frame-relay
frame-relay map novell 200.0000.0c00.7b21 102 broadcast
```

Monitoring Frame Relay

Use the EXEC commands in this section to monitor frame relay connections.

Monitoring the Frame Relay Interface

When using the frame relay encapsulation, the EXEC command **show interface** includes information on the multicast DLCI, the DLCI of the interface, and the LMI DLCI used for the local management interface.

The multicast DLCI and the local DLCI can be set using the **frame-relay multicast-dlci** and the **frame-relay local-dlci** subcommands, or provided through the local management interface. The status information is taken from the LMI, when active.

Enter this command at the EXEC prompt:

show interfaces [*type unit*]

Following is sample output.

```
Serial 2 is up, line protocol is up
  Hardware type is MCI Serial
  Internet address is 131.108.122.1, subnet mask is 255.255.255.0
  MTU 1500 bytes, BW 1544 Kbit, DLY 20000 usec, rely 255/255, load 1/255
  Encapsulation FRAME-RELAY, loopback not set, keepalive set (10 sec)
  multicast DLCI 1022,  status defined, active
  source DLCI    20, status defined, active
  LMI DLCI 1023, LMI sent 10, LMI stat recvd 10, LMI upd recvd 2
  Last input 7:21:29, output 0:00:37, output hang never
  Output queue 0/100, 0 drops; input queue 0/75, 0 drops
  Five minute input rate 0 bits/sec, 0 packets/sec
  Five minute output rate 0 bits/sec, 0 packets/sec
     47 packets input, 2656 bytes, 0 no buffer
     Received 5 broadcasts, 0 runts, 0 giants
     5 input errors, 0 CRC, 0 frame, 0 overrun, 0 ignored, 57 abort
```

```
        518 packets output, 391205 bytes
        0 output errors, 0 collisions, 0 interface resets, 0 restarts
        1 carrier transitions
```

In this display, the multicast DLCI has been changed to 1022 with the **frame-relay multicast-dlci** interface subcommand.

In this display, the statistics for the LMI are the number of status inquiry messages sent (`LMI sent`), the number of status messages received `(LMI recvd)`, and the number of status updates received (`upd recvd`). See the *Frame Relay Interface* specification for additional explanations of this output. The chapter "Configuring Interfaces" also provides explanation about the other fields seen in the **show interfaces** command.

Displaying Frame Relay Map Entries

Use this EXEC command to display the current frame relay map entries and information about these connections:

show frame-relay map

Sample output follows:

```
Serial2: IP 131.108.122.2 dlci 10(0XA,0xA0), dynamic
         status defined, active
```

The display lists the interface, the protocol, the protocol address, and the DLCI being used to reach this address. If the optional broadcast keyword was entered for a static map entry, this will also be shown.

The DLCI is displayed in three forms. For example, if the DLCI is 10, the representations would be `10(0xA,0xA0)`. The displays show the decimal value, the hexadecimal value, and the value of the DLCI as it would appear on the wire. In addition, the display indicates whether this is a static or dynamic entry. Status information for the DLCI is display if provided by the LMI.

For the CLNS protocol the map has the following form:

```
Serial0: CLNS  dlci 100(0X64,1840), static, broadcast, BW = 64000
         status defined, active
Serial0: CLNS  dlci 102(0X66,1860), static, broadcast, BW = 64000
         status defined, active
```

Displaying Global Frame Relay Statistics

Use the **show frame-relay traffic** command to display global frame relay statistics. Enter this command at the EXEC prompt:

show frame-relay traffic

Sample output follows:

```
Frame Relay statistics:
ARP requests sent 14, ARP replies sent 0
ARP request recvd 0, ARP replies recvd 10
```

```
    LMI sent 10, LMI stat recvd 10, LMI upd recvd 2, Multicast sent 48
```

Statistics for all frame relay interfaces in the router are also included in this display.

Debugging Frame Relay

Use the EXEC commands described in this section to troubleshoot and monitor activity on the interface configured for frame relay. For each **debug** command, there is a corresponding **undebug** command that turns messaging off.

debug frame-relay-events

This command enables logging of key events in the transmission or receipt of packets encapsulated using frame relay.

debug frame-relay-lmi

This command enables logging of information on the local management interface packets exchanged between the router and the frame relay service provider.

debug frame-relay-packets

This command displays all packets being sent out on the frame relay network. The display identifies the output interface, the protocol identifier, and the size of the packet being sent.

Frame Relay Interface Subcommand Summary

Following is an alphabetically arranged summary of the frame relay interface subcommands.

encapsulation frame-relay

Specifies frame relay encapsulation on a specific interface.

[**no**] **frame-relay keepalive** *number*

Enables and disables the LMI mechanism for serial lines using the frame relay encapsulation.

frame-relay local-dlci *number*

Sets the source DLCI for use when the LMI is not supported. If LMI is supported and the multicast information element is present, the network server sets its local DLCI based on information provided via the LMI. The argument *number* is the local, or source, DLCI number.

frame-relay map *protocol protocol-address DLCI* [**broadcast**]

Defines the mapping between an address and the DLCI used to connect to the address.The frame relay map tells the network server how to get from a specific protocol and address pair to the correct DLCI. The argument *protocol* can be one of these keywords: **ip**, **decnet**, **appletalk**, **xns**, **novell**, **vines**, **clns**. The keyword is followed by the corresponding protocol address and the DLCI number. The optional **broadcast** flag specifies that broadcasts should be forwarded to this address when the multicast is not enabled. The default is not to forward broadcasts.

frame-relay map clns *DLCI* **broadcast**

This variation of the **frame-relay map** command is used for the ISO CLNS protocol.

frame-relay map bridge *DLCI* **broadcast**

This variation of the **frame-relay map** command is used for bridging.

no frame-relay map

Deletes the frame relay map entry.

frame-relay multicast-dlci *number*

Defines a DLCI to be used for multicasts and should only be used when the multicast facility is *not* supported. Network transmissions (packets) sent to a multicast DLCI are delivered to all network servers defined as members of the multicast group. The argument *number* identifies the multicast group.

[**no**] **frame-relay short-status**

Instructs the network server to request the short status message from the switch (see version 2.3 of the joint *Frame Relay Interface* specification). The default is to request the full status message.

Configuring Switched Multi-Megabit Data Services (SMDS)

The Switched Multi-Megabit Data Service (SMDS) is a wide area networking service offered by Regional Bell Operating Companies (RBOCs) and other telephone service carriers such as AT&T and MCI/Sprint. The SMDS protocol is based on cell relay technology as defined in the Bellcore Technical advisories, which is in turn based on the IEEE 802.6 Standard (also called the Distributed Queue Dual Bus (DQDB) Metropolitan Area Network Media Access Control protocol). For technical references please see the bibliography in the "References and Recommended Readings" list at the end of this publication.

Cisco provides an interface to an SMDS network using DS1 transmission facilities at the rate of 1.544 Mbps. Connection to the network is made through a device called an SDSU—an SMDS CSU/DSU (Channel Service Unit/Digital Service Unit) developed jointly by Cisco Systems and Kentrox. The SDSU attaches to a Cisco router through an RS-449 connection. On the other side, the SDSU terminates a DS1 line.

Cisco's implementation of SMDS supports the IP, DECnet, AppleTalk, XNS, Novell IPX, Ungermann-Bass Net/One, and OSI internetworking protocols. Routing of IP is fully dynamic; that is, the routing tables are determined and updated dynamically. Routing of the other supported protocols requires that you establish a static routing table of SMDS neighbors in a user group. Once this is set up, all interconnected routers provide dynamic routing.

This section describes Cisco's implementation of SMDS, and how to configure, maintain, and debug SMDS.

Hardware Requirements

You need the following hardware and equipment to configure the Cisco Systems' SMDS implementation:

- An MCI or SCI serial interface controller card, or HSSI interface (chassis-based systems) or the serial port on an IGS router
- An RS-449 applique (chassis-based systems) or RS-449 transition cable (IGS)
- The SDSU device
- The packet-switched software option with the system software

Configuring SMDS

Follow these steps to configure SMDS service on the Cisco router:

Step 1: Determine the protocols you will be running over SMDS. Obtain from the service provider the group addresses that you will need to support those protocols.

Step 2: Obtain the SMDS hardware (individual) address from the service provider for each router that will interface directly into the SMDS network (that is, customer premises equipment).

Step 3: You will also need to know the addresses of the other routers with whom you'll be communicating, to set up the static routing tables. Please note that static mapping is needed only for protocols other than IP or CLNS. For IP and CLNS the routing is fully dynamic. (For more details please see these routing chapters in Part Four of this manual.)

Step 4: Configure the desired serial interface with SMDS encapsulation.

Step 5: Set up the static map for the desired protocols using the **smds static-map** command.

Step 6: Set up the multicast map for the desired protocols using the **smds multicast** command.

Using SMDS Addresses

All addresses for SMDS service are assigned by the service provider, and may be assigned to individuals and groups.

A group address (also defined as a multicast address) is entered in the Cisco SMDS configuration software using an E1 prefix; a C1 prefix is used to specify individual addresses. The Cisco software expects the addresses to be entered in a slightly modified E.164 format. E.164 format is 64 bits. The first four bits are type code followed by 10 BCD digits padded to the full 60 bits with ones.

An example of an E.164 address follows:

```
C14155561313FFFF
```

Note: To simplify the addresses, Cisco does not require the full E.164 address. The trailing FFFF's are not needed. They are not displayed and it is not necessary to type them when entering an address.

The addresses may be entered with periods in a manner similar to Ethernet-style notation, or simply as a string of digits.

An example of an individual address entered in Ethernet-style notation would look like this:

```
C141.5555.1212
```

An example of a group address would look like this:

```
E18009999999
```

Enabling SMDS

Enter the **encapsulation smds** interface subcommand to enable SMDS service on the desired interface:

encapsulation smds

The interface to which this command applies must be a serial interface. All subsequent SMDS configuration commands only apply to an interface with encapsulation SMDS.

Example:

Following is an example of the commands you use to configure the SMDS service on interface serial 0:

```
interface serial 0
encapsulation smds
```

Note: The maximum packet size allowed in the SMDS specifications (TA-772) is 9188. This is larger than what can be used by routers with most media. We therefore default the MTU to 1500 bytes, to be consistent with Ethernet. If a larger MTU is used, the **mtu** command must be used before the **encapsulation smds** command is used.

Specifying the SMDS Address

Enter the **smds address** interface subcommand to specify the SMDS individual address for a particular interface. The format of the command follows:

smds address *smds-address*
no smds address *smds-address*

Enter an individual address provided by the SMDS service provider for the argument *smds-address*. Enter the address, as described in the section "Using SMDS Addresses." This address is protocol independent.

Enter the **no smds address** command to remove the address from the configuration file.

There is no default for this command.

Example:

Following is an example which shows how a command specifies an individual address.

```
interface serial 0
smds address C141.5797.1313
```

Note: If bridging is enabled on any interface, the SMDS address is erased, and must be re-entered.

Enabling the Address Resolution Protocol

Enter the **smds enable-arp** interface subcommand to enable the Address Resolution Protocol (ARP). The full syntax of this command follows.

smds enable-arp
no smds enable-arp

The multicast address for ARP must be set before this command is issued.

By default, ARP is not enabled. Once ARP has been enabled, use the **no smds enable-arp** command to return the line to the default state.

Defining a Static Map for an Individual Address

Enter the **smds static-map** interface subcommand to configure a static mapping between an individual SMDS address and a higher level protocol address. The full syntax of the command follows:

smds static-map *protocol-type protocol-address smds-address*
no smds static-map *protocol-type protocol-address smds-address*

Do not enter this command for broadcast or multicast addresses. For those addresses, use the **smds multicast** interface subcommand described in the section "Mapping to a Multicast Address."

Enter the name of the protocol for the *protocol-type* argument, and follow with the address of the protocol to answer the *protocol-address* argument. Provide the SMDS address for the *smds-address* argument to complete the mapping. You must use these keywords to define the protocol type: **ip**, **decnet**, **appletalk**, **xns**, **novell**, or **clns**.

Use the **no smds static-map** command with the appropriate arguments to remove the map.

Example:

Following is an example of the command.

```
smds static-map XNS 111.00C0.2711.0123  C141.5688.1212
```

The command will map XNS address *111.00C0.2711.0123* to the individual SMDS address C141.5688.1212.

Mapping to a Multicast Address

Enter the **smds multicast** interface subcommand to map an SMDS group address to a broadcast or multicast address used by higher level protocols. The full syntax of the command follows:

smds multicast *protocol-type smds-address*
no smds multicast *protocol-type smds-address*

Enter the name of the protocol for the *protocol-type* argument, and follow with the SMDS address to answer the *smds-address* argument to complete the mapping. You may use these keywords to define the protocol type:

- **ip**—IP
- **arp—**ARP
- **decnet—**DECnet
- **decnet_router**—DECnet multicast address for all routers
- **decnet_node—**DECnet multicast address for all end systems
- **appletalk**—AppleTalk
- **aarp**—AppleTalk ARP address
- **xns**—XNS

- **novell**—Novell IPX
- **clns**—ISO CLNS
- **clns_is**— Multicast address for all CLNS Intermediate Systems
- **clns_es**— Multicast address for all CLNS End Systems

Since SMDS does not incorporate broadcast addressing, a group address for a particular protocol must be defined to serve the broadcast function. There is no default for this command.

Use the **no smds multicast** command with the appropriate arguments to remove a multicast address.

Example:

Following is an example of the command:

```
smds multicast IP  E180.0999.9999
```

The command maps the IP broadcast address to the SMDS group address *E180.0999.9999*.

Enabling the AT&T SMDS Service

Cisco's implementation of SMDS includes a configuration option that enables the router to interface to an AT&T SMDS router that implements an early, prestandard version of the IP encapsulation on SMDS standard protocol. This version has been superceded by the SMDS standards. However, the pre standard implementation represents an embedded base of available service. Please consult your service provider to find out if this command will be needed.

Use the **smds d15-mode** interface subcommand in order to operate with the AT&T SMDS router. The full command syntax follows:

smds d15-mode
no smds d15-mode

AT&T's implementation does not match Bellcore's SMDS specification. When this switch is enabled, the router will use AT&T's implementation. It should be off if protocols others than IP are to be routed. (In the early AT&T implementation, only IP was supported.)

AT&T modified the ARP packet structure. When this command is enabled, the router uses the AT&T D15 packet structure. The command should be disabled for systems that have not been upgraded.

Use the **no smds d15-mode** command to turn this function off. By default, the function is on.

Protocol-Specific Configuration

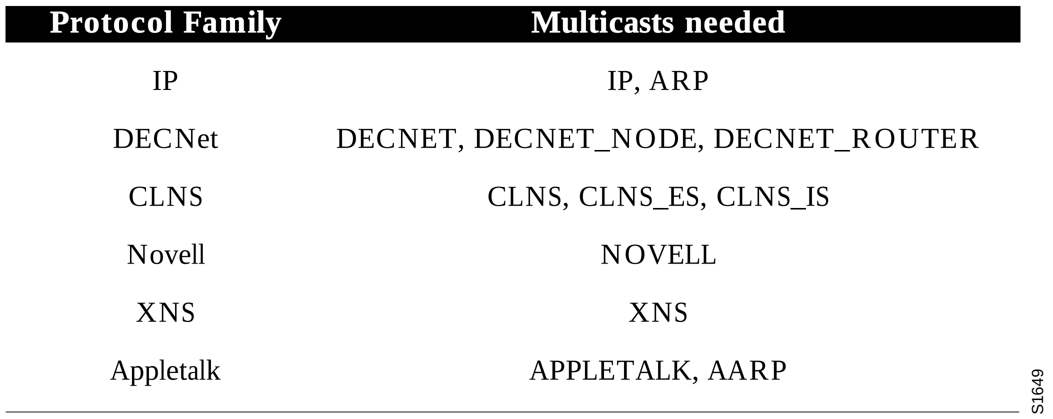
An SMDS network can be thought of in much in the same way as an X25 cloud. The premises equipment, in this case a Cisco router, represents the edge of the cloud. The service provider enables communication across the cloud. However, proper configuration is needed for communication to occur. This configuration will differ between protocol families.

One major difference between protocol families is dynamic versus static routing among the routers (called remote peers) on the periphery of the cloud. For IP and CLNS, routing across the SMDS cloud is fully dynamic. No action on the user's part is needed for the mapping of higher level protocol addresses to SMDS addresses to occur. For the other supported protocols, a static entry must be made for each of the other neighbors. This entry provides a router with the information that it needs to communicate with all other neighbor routers.

Up until now this discussion has centered on the peer routers. What about all of the end nodes and routers behind the SMDS router? The static entries only need to be made for those routers that are SMDS remote peers. Nothing additional needs to be done in order to communicate with other nodes behind the peer routers.

Some protocol families need separate definitions for associated subprotocols. The next sections illustrates how to implement these kind of configurations. See Table 1-7 for a list of protocol families and what multicast is needed.

Table 1-7 Protocol Families and the Type of Multicasts Needed



Protocol Family	Multicasts needed
IP	IP, ARP
DECNet	DECNET, DECNET_NODE, DECNET_ROUTER
CLNS	CLNS, CLNS_ES, CLNS_IS
Novell	NOVELL
XNS	XNS
Appletalk	APPLETALK, AARP

S1649

Configuring IP

Both IP and ARP should be configured with the **smds multicast** command. ARP should be enabled. The results of the ARP activity can be shown with the **show arp** command. If desired, static ARP entries may be made by using this command:

arp *ip-address address*

The argument *ip-address* is the IP address. The argument *address* is the SMDS address.

Configuring AppleTalk

Currently, dynamic address assignment does not work, and therefore an AppleTalk address must be assigned to the interface, and each remote router peer must be listed with an **smds static-map** command. The AppleTalk ARP (AARP) multicast address must still be configured with the **smds multicast** command. ARP should be enabled.

Configuring XNS and Novell

The XNS and/or NOVELL the multicast address must be configured.

For Novell, RIP Routing packets, SAP packets, NetBios Name Lookups, directed broadcasts, and traffic to the helper addresses (if that helper address is a broadcast address) will be sent to the SMDS novell multicast address.

For XNS, only RIP, directed broadcasts, and helper traffic will be sent to the XNS multicast address.

For XNS and/or Novell configurations, a static map entry must be made for each remote peer.

Configuring CLNS

Multicasts must be configured for CLNS_ES, and CLNS_IS. No static maps are necessary. ESH's, ISH's, and Router Hellos are sent to the multicast address, and neighbor entries are created automatically.

SMDS Configuration Examples

This section provides some examples of configurations to use as models for your configuration files.

Typical Multiprotocol Configuration

Following is a typical interface configured for IP, DECNET, CLNS, Novell, XNS, and AppleTalk.

Example:

```
interface Serial 4
ip address 1.1.1.2 255.0.0.0
decnet cost 4
appletalk address 92.1
appletalk zone smds
clns router igrp FOO
novell net 1a
xns net 17
encapsulation SMDS
! smds configuration follows
smds address c120.1580.4721
no smds att-mode
smds static-map APPLETALK 92.2 c120.1580.4592
smds static-map APPLETALK 92.3 c120.1580.4593
smds static-map APPLETALK 92.4 c120.1580.4594
smds static-map NOVELL 1a.0c00.0102.23ca c120.1580.4792
smds static-map XNS  17.0c00.0102.23ca c120.1580.4792
smds static-map NOVELL 1a.0c00.0102.23dd c120.1580.4728
smds static-map XNS 17.0c00.0102.23aa c120.1580.4727
smds multicast NOVELL e180.0999.9999
smds multicast XNS e180.0999.9999
smds multicast ARP e180.0999.9999
smds multicast IP e180.0999.9999
smds multicast APPLETALK e180.0999.9999
smds multicast AARP e180.0999.9999
smds multicast CLNS_IS e180.0999.9990
smds multicast CLNS_ES e180.0999.9990
smds multicast DECNET_ROUTER e180.0999.9992
smds multicast DECNET_NODE e180.0999.9992
smds enable-arp
```

Configuration with a Remote Peer on the Same Network

An example of a remote peer on the *same* SMDS network follows. Note that this router does not have DECnet routing enabled.

Example:

```
interface Serial 0
ip address 1.1.1.1 255.0.0.0
appletalk address 92.2
appletalk zone smds
clns router igrp FOO
novell net 1a
xns net 17
encapsulation SMDS
! smds configuration follows
smds address c120.1580.4792
no smds att-mode
smds static-map APPLETALK 92.1 c120.1580.4721
smds static-map APPLETALK 92.3 c120.1580.4593
smds static-map APPLETALK 92.4 c120.1580.4594
smds static-map NOVELL 1a.0c00.0102.23cb c120.1580.4721
smds static-map XNS 17.0c00.0102.23cb c120.1580.4721
smds static-map NOVELL 1a.0c00.0102.23dd c120.1580.4728
smds static-map XNS 17.0c00.0102.23aa c120.1580.4727
smds multicast NOVELL e180.0999.9999
smds multicast XNS e180.0999.9999
smds multicast ARP e180.0999.9999
smds multicast IP e180.0999.9999
smds multicast APPLETALK e180.0999.9999
smds multicast AARP e180.0999.9999
smds multicast CLNS_IS e180.0999.9990
smds multicast CLNS_ES e180.0999.9990
smds enable-arp
```

Monitoring SMDS Service

Use the following EXEC commands to monitor the SMDS service.

Displaying SMDS Individual Addresses

Use the **show smds addresses** command to display the individual addresses and the interface that they are associated with. Enter this command at the EXEC prompt:

show smds addresses

A sample display of the command output follows:

```
Artemis#show smds addresses
SMDS address - Serial0   c141.5555.1212
```

Displaying Mapped SMDS Addresses

Use the **show smds map** command to display all SMDS addresses that are mapped to higher level protocol addresses. Enter this command at the EXEC prompt:

show smds map

The display for this command includes all addresses entered with both the **smds static-map** and **smds multicast** command.

A sample display of the command output follows:

```
Artemis#show smds map
Serial0:  ARP maps to e180.0999.9999 multicast
Serial0:  IP maps to e180.0999.9999 multicast
Serial0:  XNS 1006.AA00.0400.0C55 maps to c141.5688.1212 static
```

Note: Trailing Fs are implied in displays showing the SMDS addresses.

Displaying SMDS Counters

Use the **show smds traffic** command to display all the SMDS counters. Enter this command at the EXEC prompt:

show smds traffic

A sample display of the command output follows:

```
Artemis#show smds traffic
0 Bad BA size errors
0 Bad Header extension errors
0 Invalid address errors
```

In the display:

- The Bad BA size errors field lists the number of corrupted packets received based on the expected Level 3 PDU buffer allocation size.

- The Bad Header extension errors field lists the number of invalid packets received in which the header extension length did not match what was expected in the Level 3 PDU.
- The Invalid address errors field lists the number of packets passed that were incorrectly sent to router. Both individual and multicast address errors are included in this count.

Debugging SMDS

Use the following EXEC command to debug the SMDS service. For each **debug** command, there is a corresponding **undebug** command that turns the messages off.

debug arp

Use this command to see if ARPs are being sent or received. This command prints one line for each ARP sent or received.

debug serial-interface

Use this EXEC command to enable logging of SMDS events. This command prints a one-line message for each packet that is sent. The packet size, packet type, and SMDS source and destination addresses is printed. If packets are received with an incorrect destination address, it will be noted and counted.

SMDS Interface Subcommand Summary

This section provides an alphabetically arranged summary of the SMDS interface subcommands. These commands must be preceded by an **interface** command.

encapsulation smds

Enables or disables SMDS on a particular interface. It should precede any SMDS command. This command has no default.

[**no**] **smds address** *smds-address*

Sets or removes the SMDS individual address for a particular interface. The argument *smds-address* is the individual address provided by the SMDS service provider, and is protocol independent. This command has no default.

[**no**] **smds att-mode**

Enables or disables AT&T's early prestandard SMDS implementation. Default is **smds att-mode**. This command is on by default.

[**no**] **smds enable-arp**

Enables or disables ARP processing on a particular interface. The multicast address for ARP must be set before this command is issued. Default is **no smds enable-arp**.

[**no**] **smds multicast protocol-type smds-group-address**

Maps an SMDS group address to a broadcast or multicast address used by higher-level protocols. This command has no default.

[**no**] **smds static-map protocol-type protocol-address smds-address**

Sets up a static mapping between an SMDS address and a higher-level protocol address. This should not be used for broadcast addresses, for broadcast address, use the **smds multicast** command. This command has no default.