DECbrouter 90 Products

Configuration and Reference Volume 3

Order Number: EK–DECB3–CG. A01

First Edition, May 1993

This document was prepared using VAX DOCUMENT, Version 2.1.

Contents

Figures

Tables

About This Guide

This section discusses the objectives, audience, organization, and conventions of this guide. It also lists related Digital publications.

Audience

This publication is intended for system administrators who will configure and maintain a DECbrouter 90.

Organization

The following chapters are contained in this guide:

- **Chapter 1, Switching ISO CLNS**–provides an overview of basic ISO terminology, and describes CLNS address assignment, configurations for basic ISO CLNS, and switching features.
- **Chapter 2, Configuring ISO CLNS Routing Protocols**–describes routing protocols supported by the International Organization for Standardization (ISO) Connectionless Network Services (CLNS) protocol.
- **Chapter 3, Routing Novell IPX**–describes the DECbrouter 90 implementation of the Novell IPX routing protocol.
- **Chapter 4, Routing PUP**–describes routing configuration using the Xerox PARC Universal Protocol (PUP). Monitoring and debugging commands are also described.
- **Chapter 5, Routing VINES**–describes the DECbrouter 90 implementation of the Banyan VINES routing protocol.
- **Chapter 6, Routing XNS**–describes how to configure your router for packets using the Xerox Network System (XNS) protocols.
- **Chapter 7, Configuring Transparent Bridging**–describes the transparent bridging support available in the DECbrouter 90.
- **Chapter 8, Configuring Serial Tunneling and SDLC Transport**–describes the implementation of the DECbrouter 90 serial tunneling (STUN) and SDLC transport.
- **Chapter 9, LLC2 and SDLC Link-Level Support**–provides an overview of the implementation of the two IBM data link level protocols.
- **Chapter 10, Fast Switching**–provides information on protocols and interfaces that support fast switching.
- **Chapter 11, Point-to-Point Protocol (PPP) for DECnet IV and OSI**–describes how to configure the PPP, and enable DECnet and OSI routing.
- **Appendix A, System Error Messages**–lists the system error messages.

- **Appendix B, Access List Summary**–summarizes the general command syntax and number ranges or symbolic names, used for the access lists supported by the DECbrouter 90T software.
- **Appendix C, Ethernet Type Codes**–lists the Ethernet type codes for use with type code filtering configuration commands.
- **Appendix D, Pattern Matching**–describes the pattern-matching scheme used by the X.25 switching feature.
- **Appendix E, ASCII Character Set**–lists the ASCII character set.
- **Appendix F, References and Recommended Reading**–lists reference material and recommended reading.
- **Appendix G, X.25 Diagnostic Codes**–contains information about the X.25 diagnostic codes.

Conventions

The following conventions are used in this guide:

Convention	Meaning
`system displays`	Terminal sessions and information the system displays are printed in monospace type.
boldface	Information you enter is in boldface type. Keywords are also in boldface type.
>	Nonprinting characters are shown in angle brackets.
[ ]	Defaults are in square brackets.
{ }	Required keywords are in braces { } and are separated by a vertical bar (|).
COMMANDS	Commands are in uppercase letters.
italics	Command variables, worksheet values, new terms and concepts, and titles of books and periodicals are in italics.
Ctrl/X	Indicates two keys that you must press simultaneously.
Public	A string should not be enclosed by quotes (""). For example, when setting up a community string for SNMP to "public," do not enclose the word public with quotes; otherwise, the string would be set to "public".
Note	Provides general information about the current topic.
Caution	Provides information to prevent damage to equipment.

Related Documentation

The most important companion to this guide is the set of Digital documentation that accompanies the product. This guide refers you to the appropriate manual for additional reference material that is beyond the scope of this guide. The following manuals are shipped with the DECbrouter 90:

- *DECbrouter 90 Products Getting Started*
- *DECbrouter 90 Products Configuration and Reference, Volume 1*
- *DECbrouter 90 Products Configuration and Reference, Volume 2*
- *DECbrouter 90T Installation and Operating Information*
- *DECbrouter 90 System Error Messages*
- *DECbrouter 90 Products Command Summary*

To order these publications or additional copies of this guide, contact your sales representative. Refer to the How to Order Additional Documentation page at the back of this document for addresses and phone numbers.

Service and Support

Call your local representative for service.

Obtaining Additional Information

This section describes how to obtain additional Digital publications and includes tips for obtaining books, standards, and other information about networks and data communications that might be helpful while using Digital products.

Ordering Additional Digital Publications

To order these publications or additional copies of the *DECbrouter 90 Products Configuration and Reference* guides, contact your local representative.

Obtaining Information

This section describes how to obtain RFCs and technical standards.

Obtaining RFCs

Information about the Internet suite of protocols is contained in documents called *Requests for Comments or RFCs*. These documents are maintained by Government Systems, Inc. (GSI). You can request copies by contacting GSI directly, or you can use the TCP/IP File Transfer Protocol (FTP) to obtain an electronic copy.

Contacting GSI

You can contact GSI through mail, by telephone, or through electronic mail:

Government Systems, Incorporated
Attn: Network Information Center
14200 Park Meadow Drive, Suite 200
Chantilly, Virginia 22021

1-800-365-3642
(703) 802-4535
(703) 802-8376 (FAX)

`NIC@NIC.DDN.MIL`

Network address: 192.112.36.5
Root domain server: 192.112.36.4

Obtaining an Electronic Copy

To obtain an electronic copy of an RFC through FTP, complete the following step

1. At your server prompt, use the **ftp** command to connect to address *nic.ddn.mil*:

   ```
   % ftp nic.ddn.mil
   ```

The following display appears, followed by a login prompt:

```
Connected to nic.ddn.mil.
220-*****Welcome to the Network Information Center*****
    *****Login with username "anonymous" and password "guest"
    *****You may change directories to the following:
      ddn-news          - DDN Management Bulletins
      domain            - Root Domain Zone Files
      ien               - Internet Engineering Notes
      iesg              - IETF Steering Group
      ietf              - Internet Engineering Task Force
                            internet-drafts   - Internet Drafts
                          netinfo           - NIC Information Files
      netprog           - Guest Software (ex. whois.c)
      protocols         - TCP-IP & OSI Documents
      rfc               - RFC Repository
      scc               - DDN Security Bulletins
220 And more.
```

2. At the login prompt, enter the word **anonymous** as your login name:

   ```
   Name (nic.ddn.mil:cindy): anonymous
   ```

   The NIC responds with this message:

   ```
   331 Guest login ok, send "guest" as password.
   Password:
   ```

3. Enter the word **guest** at the Password: prompt. The following message and ftp> prompt appear:

   ```
   230 Guest login ok, access restrictions apply.
   ftp>
   ```

4. Use the **cd** command to change directories. The following example illustrates how to change the RFC directory and obtain RFC 1158:

   ```
   ftp> cd rfc
   250 CWD command successful.
   ftp> get rfc1158.txt
   ```

5. To exit the FTP facility, enter the **quit** command at the ftp> prompt.

Obtaining Technical Standards

Following are additional sources for technical standards:

- Omnicom, 1-800-OMNICOM
- Global Engineering Documents, 2805 McGraw Ave., Irvine, CA 92714
  1-800-854-7179
- American National Standards Institute, 1430 Broadway, New York, NY 10018
  (212) 642-4932 or (212) 302-1286

1 Switching ISO CLNS

This publication includes two chapters that address the International Organization for Standardization (ISO) Connectionless Network Services (CLNS) protocol, which is a standard for the network layer of the Open Systems Interconnection (OSI) model. This chapter focuses on the following topics:

- Overview of basic ISO terminology
- CLNS address assignment
- Basic ISO CLNS configuration
- Configuration of switching features
- Configuration examples related to topics addressed in this chapter (including command listing and illustrations)

Commands are summarized at the end of this chapter. See Chapter 2 for information on the various routing protocols and how they are used in internetworks.

The DECbrouter 90 Implementation of ISO CLNS

The DECbrouter 90 routing software supports packet forwarding and routing for ISO CLNS on networks using Ethernet and serial data link layers.

You can use CLNS routers on serial interfaces with HDLC, LAPB, X.25, SMDS, or Frame Relay encapsulation. To use HDLC encapsulation, you must have a router at both ends of the link. If you use X.25 encapsulation, you must manually enter the NSAP-to-X.121 mapping. The LAPB, SMDS, Frame Relay, and X.25 encapsulations will interoperate with other vendors.

In addition, this CLNS implementation is compliant with the Government Open Systems Interconnection Profile (GOSIP) Version 2. As part of its CLNS support, the DECbrouter 90 fully supports these ISO and ANSI standards:

- ISO 9542—Documents the End System-to-Intermediate System (ES-IS) routing exchange protocol
- ISO 8473—Documents the ISO Connectionless Network Protocol (CLNP)
- ISO 8348/Ad2—Documents Network Service Access Points (NSAPs) ISO 10589—Documents Intermediate System-to-Intermediate System (IS-IS) Intra-domain Routing Exchange Protocol

Both the ISO-developed IS-IS and Cisco's ISO Interior Gateway Routing Protocol (ISO-IGRP) dynamic routing protocols are supported for dynamic routing of ISO CLNS.

In addition, static routing for ISO CLNS is supported.

Support for IS-IS, ISO-IGRP, and static routing are described in Chapter 2.

Switching ISO CLNS

In this publication, *switching* involves taking a packet in one interface and sending it out another (or the same) interface. Switching decisions always are made by looking in a routing table. If the routing table is built dynamically, the process that does so implements a *routing* protocol. If the routing table is built by user configuration, the router is doing *static routing*.

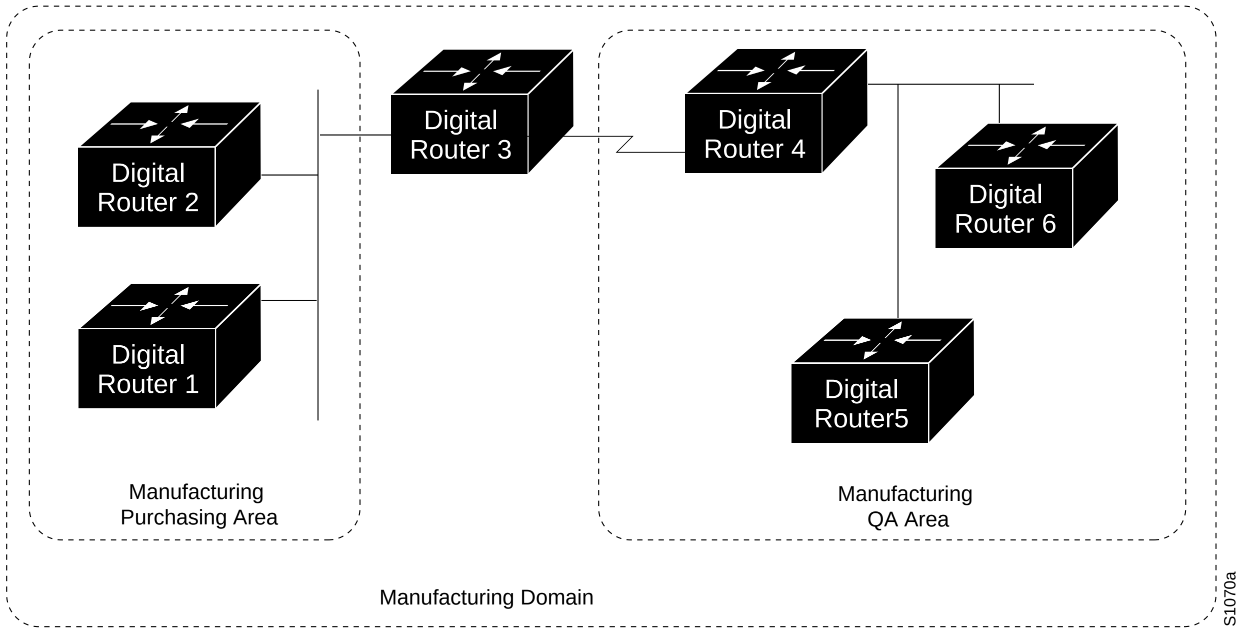
When configured for switching only, the router only makes forwarding decisions. It does not perform other routing-related functions. In such a configuration, the router compiles a table of adjacency data, but does not advertise this information. The only information that is inserted into the routing table is the NSAP and Network Entity Title (NET) addresses of this router, static routes, and adjacency information.

ISO Routing Terminology

The ISO network architecture uses terminology and concepts that differ from IP networks. In other network architectures, CLNS would be known as a *datagram service*; in the TCP/IP architecture, for example, datagram service is provided by IP.

The lowest level of the routing hierarchy is the *area*—a set of end systems connected by intermediate systems (routers). (See Figure 1–1). The network itself has no separate identity, no network number; instead, the next addressable entity above the end system or intermediate system (router) is the area.

Figure 1–1 ISO CLNS Areas



Areas are connected to other areas to form *routing domains*. Each domain is a separately administered region, similar in concept to an autonomous system in IP networks.

Some intermediate systems keep track of how to communicate with all of the end systems in their areas and thereby function as local routers (sometimes referred to as Level 1 routers). Other intermediate systems keep track of how to communicate with other areas in the domain, functioning as area routers (sometimes referred to as Level 2 routers). DECbrouter 90 routers are always local and area routers when routing ISO-IGRP; they can be configured to be local only, area only, or both local and area routers when routing IS-IS.

End systems communicate with intermediate systems using the ES-IS protocol. Level 1 and Level 2 intermediate systems communicate with each other using either ISO IS-IS or Digital's ISO-IGRP protocol.

Routing across domains (interdomain routing) can be done either statically or dynamically (with ISO-IGRP).

Specific ISO CLNS routing protocols supported by the DECbrouter 90 are discussed in Chapter 2.

Configuring ISO CLNS

Follow these steps to configure your router to support ISO CLNS:

1. Enable CLNS routing with the CLNS ROUTING global configuration command.
2. Create dynamic routing processes with the ROUTER global configuration command and either the **iso-igrp** or **isis** keywords. You do not need to explicitly specify a routing process to use static routing facilities. Refer to Chapter 2, for more information about static and dynamic routing.
3. Specify which interface should be actively routing ISO CLNS with the CLNS ROUTER interface subcommand.

   If you do not intend to use a dynamic routing protocol on a specific interface, but wish to switch packets over the link, use the CLNS ENABLE interface subcommand.

These steps enable CLNS routing. Optional commands are also available for customizing the environment, and you can follow these steps to complete configuration:

1. Configure static information, such as prefix routes and/or NSAP to media address mappings.
2. Adjust ES-IS parameters, as necessary.

Each task is described in the following sections or in Chapter 2 (routing protocol specifics). Configuration information is followed by descriptions of the EXEC commands to maintain, monitor and debug the ISO CLNS network. Summaries of the global configuration commands and interface subcommands described in these sections appear at the end of this chapter.

CLNS Addresses

Addresses in the ISO network architecture are referred to as Network Entity Titles (NETs) and Network Service Access Points (NSAPs). Each node in an ISO network has one or more NET. In addition, each node has many NSAPs. Each NSAP differs from one of the NETs for that node in only the last byte. This byte is called the *n-selector*. (See Figure 1–2.) Its function is similar to the port number in other protocol suites.

The DECbrouter 90 implementation supports all NSAPs that are defined by ISO 8348/Ad2; however, Digital only provides dynamic routing (ISO-IGRP or IS-IS routing), for NSAPs that conform to the address constraints defined in the ISO standard for IS-IS (ISO 10589). Figure 1–2 and Figure 1–3 illustrate the conforming NSAP addresses.

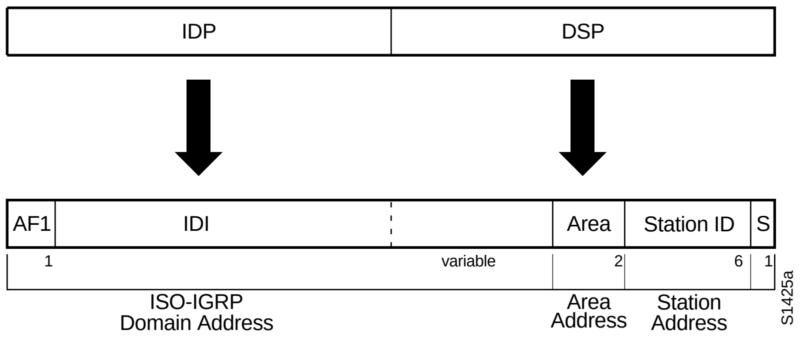
An NSAP consists of two major fields: the IDP and the DSP.

- The IDP is made up of 1-byte AFI and a variable length IDI. The length of the IDI and the encoding format for the DSP is based on the value of the AFI.
- The DSP is made up of a high-order DSP, an area ID, a system ID, and a 1-byte n-selector.

The key difference between the ISO-IGRP and IS-IS NSAP addressing schemes is in the definition of area addresses. Both use the system ID for Level 1 routing.

However, they differ in the way addresses are specified for area routing. An ISO-IGRP NSAP address includes three separate levels for routing: the *domain*, *area*, and, *system* ID. An IS-IS address includes two fields: a single continuous area field comprising the domain and area fields defined for ISO-IGRP and the *system* ID.

Figure 1–2 ISO-IGRP NSAP Addressing Structure



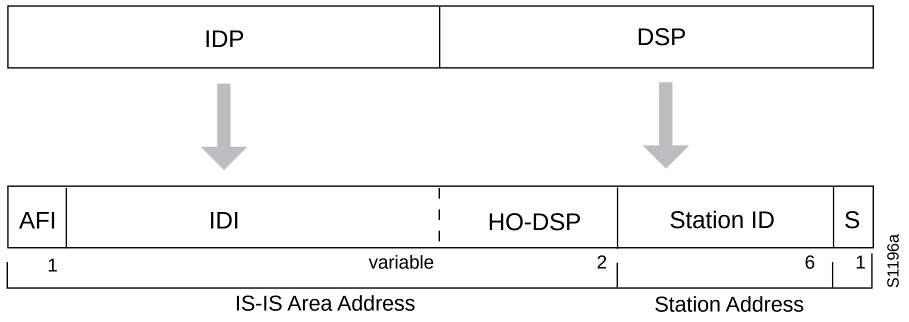
The ISO-IGRP NSAP is divided into three parts: a domain part, an area address, and a system ID. Domain routing is performed on the domain part of the address. Area routing for a given domain uses the area address. System ID routing for a given area uses the system ID part. The NSAP is laid out as follows:

- The n-selector is the last byte of the NSAP address.
- The system ID is the six bytes before the n-selector.
- The area address is the two bytes before the system ID.
- The domain part is of variable length and comes before the area address.

The DECbrouter 90 ISO-IGRP CLNS routing implementation interprets the bytes from the AFI up to (but not including) the Area field in the DSP as a *domain identifier*. The Area field specifies the *area*, and the system ID specifies the *system*.

The domain address uniquely identifies the routing domain. Within each routing domain, you can set up one or more areas. The area address uniquely identifies the area.

Figure 1–3 IS-IS NSAP Addressing Structure



An IS-IS NSAP is divided into two parts: an area address (AA) and a system ID. Level-2 routing uses the AA. Level-1 routing uses the system ID address. The NSAP is laid out as follows:

- The n-selector (S) is the last byte of the NSAP address.
- The system ID is found between the area address and the n-selector byte.
- The area address is the NSAP address, not including the system ID and n-selector.

The DECbrouter 90 ISO IS-IS CLNS routing implementation interprets the bytes from the AFI up to (but not including) the system ID field in the DSP as an *area identifier*. The system ID specifies the *system*.

The area address uniquely identifies the routing area. Within each routing domain you can set up one or more areas.

Addressing Rules

All NSAPs must obey the following constraints:

- No two nodes can have addresses with the same NET; that is, addresses that match all but the n-selector (S) field in the DSP.
- ISO-IGRP requires at least 10 bytes of length; one for domain, two for area, six for system ID, and one for n-selector.
- No two nodes residing within the same area can have addresses in which the system ID fields are the same.

Examples

The following are examples of OSInet and GOSIP NSAPs using the ISO-IGRP implementation:

OSInet:

```
47.0004.004D.0003.0000.0C00.62E6.00
|     Domain|Area|    System ID| S|
```

The following is the GOSIP NSAP Format for NSFNet and is an example of the GOSIP NSAP address structure. Structure is mandatory for addresses allocated from the International Code Designator (ICD) 0005 addressing domain. Refer to the GOSIP document, *U.S. Government Open Systems Interconnection Profile (GOSIP)*, Draft Version 2.0, April 1989 for more information.

```
 |          Domain          |
 47.0005.80.ffff00.0000.ffff.0004.0000.0c00.62e6.00
 |   |    |    |    |    |    |           |      |
 |   |    |    |    |    |    |           |      |
AFI IDI  DFI  AAI  Resv  RD  Area     System ID  N-selector
```

Multihoming in IS-IS Areas

The area addressing scheme allowed in IS-IS routing supports assignment of multiple area addresses. This concept is referred to as *multihoming*. Multiple area addresses are assigned statically on the router with the NET router subcommand defined in Chapter 2. It is recommended that all of the addresses have the same system ID.

The DECbrouter 90 currently supports assignment of up to three area addresses for a given area. The number of areas allowed in a domain is unlimited.

Multihoming provides a mechanism for smoothly migrating network addresses:

- Splitting up an area—Stations within a given area can accumulate to a point that they are difficult to manage, cause excessive traffic, or threaten to exceed the usable address space for an area. Multiple area addresses can be assigned so that you can smoothly partition a network into separate areas without disrupting service.
- Merging areas—Use transitional area addresses to merge as many as three separate areas into a single area that share a common area address.
- Transition to a different address—You may need to change an area address for a particular group of stations. Use multiple area addresses to allow incoming traffic intended for an old area address to continue being routed to associated stations.

A router can dynamically learn about any adjacent router. As part of this process, the routers inform each other of their area addresses. If two routers share at least one area address, the set of area addresses of the two routers are merged. The merged set cannot contain more than three addresses. If there are more than three, the three addresses with the lowest numerical values are kept, and all others are dropped.

Example

The following example NET commands illustrate assignment of three separate area addresses for a single router:

```
router isis eng-area1
net 47.0004.004D.0001.0000.0C00.1111.00
net 47.0004.004D.0002.0000.0C00.1111.00
net 47.0004.004D.0003.0000.0C00.1111.00
!  |    IS-IS Area  |    System ID| S|
```

The result is that traffic received that includes an area address of 47.0004.004D.0001, 47.0004.004D.0002, or 47.0004.004D.0003 and that has the same system ID, will be forwarded to the destination station in *eng-area1*.

Enabling CLNS Routing

Conceptually, each End System (ES) lives in one area. It discovers the nearest Level 1 IS (router) by listening to ES-IS packets. Each ES must be able to communicate directly with a Level 1 IS in its area.

When an ES wants to communicate with another ES, it sends the packet to the Level 1 IS for its area. The IS will look up the destination NSAP and forward the packet along the best route. If the destination NSAP is for an ES in another area, the Level 1 IS will send the packet to the nearest Level 2 IS. The Level 2 IS will forward the packet along the best path for the destination area until it gets to a Level 2 IS that is in the destination area. This IS will forward the packet along the best path inside the area until it is delivered to the destination ES.

Configuring CLNS for the Router

The configuration process begins with enabling CLNS routing. Enable CLNS routing with the CLNS ROUTING global configuration command. The full syntax for this command follows.

clns routing
no clns routing

Use the NO CLNS ROUTING command to disable CLNS routing.

You need to decide now whether you want to use static or dynamic routing. Static routing is required if you must interoperate with another vendor's equipment that does not support IS-IS. In addition, static routing is generally preferable for operation over X.25, Frame Relay, or SMDS networks.

Note

ISO-IGRP is the only dynamic routing process that you can use over X.25, Frame Relay, or SMDS networks.

You also must enable ISO CLNS for each interface. This is done automatically when configuring IS-IS or ISO-IGRP routing on an interface; however, if you do not intend to perform any dynamic routing on an interface, but intend to pass ISO CLNS packet traffic to end systems, use the CLNS ENABLE interface subcommand. The syntax for this command is as follows:

clns routing
no clns enable

Use the NO CLNS ENABLE command to disable ISO CLNS on a particular interface.

Configuring CLNS over X.25

X.25 is not a broadcast medium, and therefore ES-IS generally is not used to automatically advertise and record NSAP/NET (protocol address) to SNPA (media address) mappings. Operation of CLNS over X.25 requires that this mapping information be statically entered using the CLNS IS-NEIGHBOR and/or the CLNS ES-NEIGHBOR interface subcommands.

Configuring a serial line to use CLNS over X.25 requires configuring the general X.25 information and the CLNS-specific information. The general X.25 information must be configured first. Then, the CLNS IS-NEIGHBOR and CLNS ES-NEIGHBOR interface subcommands are issued to list all the Intermediate and End Systems that will be used. Full command syntax for these commands follows.

clns es-neighbor *nsap snpa* [*X.25-facilities-info*]
no clns es-neighbor *nsap*

clns is-neighbor *nsap snpa* [*X.25-facilities-info*]
no clns is-neighbor *nsap*

In this case, the Subnetwork Points of Attachment (SNPAs) are the X.25 network addresses (X.121 addresses). These are usually assigned by the X.25 network provider. Use the argument *X.25-facilities-info* to specify nondefault packet and window size, reverse charge information, and so on. Refer to the *DECbrouter 90 Products Configuration and Reference, Volume 1* Chapter 8, for more information.

Example 1

This command maps NSAP 47.0004.004d.0001.3132.3334.3536.00 to X.121 address 310117.

```
clns es-neighbor 47.0004.004d.0001.3132.3334.3536.00 310117
```

Only one **es-neighbor** or one **is-neighbor** entry can be made for each X.121 address.

The X.25 facilities information that can be specified is exactly the same as in the X25 MAP subcommand described in the *DECbrouter 90 Products Configuration and Reference, Volume 1*, Chapter 8.

You can specify the following information:

- Packet window size (send and receive) for mapping
- Packet size (send and receive)
- Reverse charges requested
- Reverse charges accepted
- Closed user group
- Maximum number of virtual circuits (VCs) to open
- Broadcasts
- Transit delay
- Throughput or bandwidth

- Module size
- Network user ID (nuid) support

Note

Using the *broadcast* option is only useful if you are using ISO-IGRP. Turning on broadcast tells the router to send out dynamic routing information over this interface to all configured neighbors.

Example 2

Following is a more complicated example that specifies nondefault packet and window size:

```
clns is-neighbor 47.0004.004d.0001.3132.3344.3535.00 310117 windowsize 7 7
packetsize 512 512
```

Note

This configuration command must be given on one line.

The system does not accept an X.25 call unless the source of the call has been configured as an ES or IS neighbor in the destination router. The system will not accept a call requesting reverse charges unless the keyword **accept-reverse** is used as follows.

```
clns is-neighbor 47.0004.004d.0001.1112.1314.1516.00 310120249 accept-reverse
```

All of the information that is entered using the CLNS IS-NEIGHBOR and CLNS ES-NEIGHBOR subcommands actually goes into two places in the system: the adjacency table and the X.25 map table. The adjacency table stores only the NSAP/NET and X.121 address information. The X.25 map table stores this information but also stores the facility information. If a virtual circuit (VC) has been established, the logical channel numbers (LCNs) in use are shown.

Configuring ES-IS Parameters

This section describes the commands used to configure the ES-IS parameters for host-router communication. In general, however, these should be left at their default values.

When configuring an ES-IS router, be aware of the following:

- ES-IS does not run over X.25 links unless the *broadcast* facility is enabled.
- ES Hello (ESH) packets and IS Hello (ISH) packets are sent without options. Options in received ESH and ISH packets are ignored.

Specifying Hello Packets

The CLNS CONFIGURATION-TIME global configuration command specifies the rate at which ESH and ISH packets are sent.

clns configuration-time *seconds*
no clns configuration-time

The default value for *seconds* is 60. This value is restored by the NO CLNS CONFIGURATION-TIME command.

The CLNS HOLDING-TIME global configuration command allows the sender of an ESH or ISH to specify the length of time during which the information in the Hello packets will be believed.

clns holding-time *seconds*
no clns holding-time

The argument *seconds* specifies the time in seconds. The default value is 300 seconds (five minutes), which is restored by the NO CLNS HOLDING-TIME command.

Configuring Static Configuration of ESs

End systems need to know how to get to a Level 1 IS for their area, and Level 1 ISes need to know all of the ESes that are directly reachable through each of their interfaces. To provide this information, the routers support the ES-IS protocol. A router dynamically discovers all ESes running the ES-IS protocol. ESes that are not running the ES-IS protocol must be statically configured. This is done using the CLNS ES-NEIGHBOR interface subcommand:

clns es-neighbor *nsap snpa*
no clns es-neighbor *nsap*

The argument *nsap* specifies the CLNS address. The argument *snpa* specifies the data link address. The **no** keyword deletes the ES neighbor. If you have configured the CLNS ROUTER ISO-IGRP or CLNS ROUTER ISIS subcommand for a particular interface, the ES-IS routing software automatically turns ES-IS on for that interface.

It is sometimes desirable for a router to have a neighbor entry statically configured rather than learned through ES-IS, ISO-IGRP or IS-IS. The CLNS IS-NEIGHBOR interface subcommand enters an IS neighbor.

clns is-neighbor *nsap snpa*
no clns is-neighbor *nsap*

The argument *nsap* specifies the NSAP address. The argument *snpa* specifies the data link address. The NO CLNS IS-NEIGHBOR command deletes the specified IS neighbor.

If there are systems on the Ethernet that do not use ES-IS, or if X.25 is being used, NSAP/NET (protocol address) to SNPA (media address) mappings must be specified using the CLNS IS-NEIGHBOR and/or the CLNS ES-NEIGHBOR interface subcommands.

Example

An example of configuring an Ethernet interface with the Ethernet address (MAC address) of systems that do not use ES-IS follows.

```
interface ethernet 0
clns es-neighbor 47.0004.004D.0055.0000.0C00.A45B.00  0000.0C00.A45B
```

In this case, the end systems with the following NSAP (or NET) is configured at an Ethernet MAC address of 0000.0C00.A45B:

```
47.0004.004D.0055.0000.0C00.A45B.00
```

Note

It is only necessary to use static mapping for those end systems that do *not* support ES-IS. The router will continue to dynamically discover those end systems that *do* support ES-IS.

Configuring Performance Parameters

Generally, you do not need to change the default settings for CLNS packet switching, but there are some modifications you can make when you decide that it is advantageous.

Specifying the MTU Size

All interfaces have a default maximum packet size. You can set the maximum transmission unit (MTU) size of the packets sent on the interface using the CLNS MTU interface subcommand. The full syntax of this command follows.

clns mtu size
no clns mtu

The minimum value for the size argument is 512; the default and maximum packet size depends on the interface type. The default value is restored by the NO CLNS MTU command.

Note

Changing the MTU value with the MTU interface subcommand (described in Chapter 7 of this manual) can affect the value for the CLNS MTU interface subcommand. If the current value specified with the CLNS MTU interface subcommands is the same as the value specified with the MTU interface subcommand, then when you change the value for the MTU interface subcommand, the value for CLNS MTU is automatically modified to match the new MTU interface subcommand value. However, the reverse is not true. In other words, changing the value for the CLNS MTU subcommand has no effect on the value for the MTU interface subcommand.

Configuring Checksums

When the ISO CLNS routing software sources a CLNS packet, by default it generates checksums. The CLNS CHECKSUM interface subcommand specifies this function. Use the NO CLNS CHECKSUM command to disable checksum generation.

clns checksum
no clns checksum

Note

This command has no effect on routing packets (ES-IS, ISO-IGRP and IS-IS) sourced by the system. It applies to pings and trace route packets.

Enabling Fast Switching

The CLNS ROUTE-CACHE interface subcommand allows fast switching through the cache and by default, is enabled. To disable fast switching, use the **no clns route-cache** command.

clns route-cache
no clns route-cache

Note

The cache still exists and is used after the NO CLNS ROUTE-CACHE command is used; the software just does not do fast switching through the cache.

Setting the Congestion Threshold

If a router configured for CLNS experiences congestion, it sets the congestion experienced bit. The congestion threshold is a per-interface parameter set by the CLNS CONGESTION-THRESHOLD interface subcommand. The full syntax for this command follows.

clns congestion-threshold *number*
no clns congestion-threshold

This subcommand causes the system to set the congestion-experience bit if the output queue has more than the specified number of packets in it. A *number* value of zero or the **no** keyword prevents this bit from being set. The default value for *number* is 4.

Use the NO CLNS CONGESTION-THRESHOLD command with the appropriate value to remove the parameter setting.

Transmitting Error PDUs

When a CLNS packet comes in, the routing software looks in the routing table for the next hop. If it does not find the next hop, the packet is discarded and an error Protocol Data Unit (ERPDU) may be sent.

Sending an Error PDU

The CLNS SEND-ERPDU interface subcommand allows CLNS to send an error PDU when it detects an error in a data PDU, and by default, is enabled. To disable this function, use the NO CLNS SEND-ERPDU command. The syntax for both commands follows.

clns send-erpdu
no clns send-erpdu

Determining the Interval Between ERPDUs

The CLNS ERPDU-INTERVAL interface subcommand determines the minimum interval time, in milliseconds, between ERPDUs. The full syntax of this command follows.

clns erpdu-interval *milliseconds*
no clns erpdu-interval *milliseconds*

A *milliseconds* value of zero or the NO CLNS ERPDU-INTERVAL command turns off the interval rate and effectively sets no limit to the ERPDU rate. The default rate is once every ten milliseconds.

The CLNS ERPDU-INTERVAL subcommand will not send ERPDUs more frequently than one per interface per ten milliseconds.

Redirecting PDUs

If a packet is sent out the same interface it came in on, a redirect PDU (RDPDU) also may be sent to the sender of the packet. The CLNS SEND-RDPDU interface subcommand allows CLNS to send redirect PDUs when a better route for a given host is known; this is the default behavior. The full syntax of the command follows.

clns send-rdpdu
no clns send-rdpdu

To disable this function, use the NO CLNS SEND-RDPDU command.

Disabling RDPDUs

An RDPDU can be disabled on a per-interface basis using the CLNS RDPDU-INTERVAL interface subcommand. The full syntax of the command follows.

clns rdpdu-interval *milliseconds*
no clns rdpdu-interval *milliseconds*

The command determines the minimum interval time, in milliseconds, between RDPDUs. A *milliseconds* value of zero or the NO CLNS RDPDU-INTERVAL command turns off the interval rate and effectively sets no limit to the RDPDU rate. The default rate is once every 100 milliseconds. An RDPDU is rated-limited and is not sent more frequently than one per interface per 100 milliseconds.

Disabling the RDPDU Mask

The address mask is normally present on all RDPDUs, but can be disabled with the NO CLNS RDPDU-MASK interface subcommand. The full syntax of the CLNS RDPDU-MASK command follows.

clns rdpdu-mask
no clns rdpdu-mask

Note

SNPA masks are never sent, and RDPDUs are ignored by the router when the router is acting as an IS.

Configuring Parameters for Locally Sourced Packets

Use these commands to configure parameters for packets sourced by this router. Full command syntax for each command is listed with the descriptions. The **no** forms of the commands remove the parameter's settings.

The CLNS PACKET-LIFETIME global configuration command specifies the initial lifetime for locally generated packets.

clns packet-lifetime *number*
no clns packet-lifetime

The default value for *number* is 64.

The CLNS WANT-ERPDU interface subcommand specifies whether to request error PDUs on packets sourced by the router.

clns want-erpdu
no clns want-erpdu

The default is to request error PDUs.

Note

This command has no effect on routing packets (ES-IS, ISO-IGRP and IS-IS) sourced by the system. It applies to pings and trace route packets.

Header Options

The ISO CLNS routing software ignores the Record Route option, the Source Route option, and the QOS (quality of service) option other than congestion experienced. The security option causes a packet to be rejected with a bad option indication.

NSAP Shortcut Command

The CLNS HOST global configuration command can be useful for creating a name for an NSAP. This name can then be used in place of typing the long set of numbers associated with an NSAP. The command syntax follows.

clns host *name nsap*

The argument *name* is the desired name for the NSAP, which is specified by the *nsap* argument. The assigned name is displayed, where applicable, in show and debug commands.

There are some effects and requirements associated with using names to represent NETs and NSAPs, however. Although using names as proxies for addresses is allowed with CLNS commands, the name is never written out to NVRAM.

The CLNS HOST command is generated after all other CLNS commands when the configuration file is parsed. As a result, the NVRAM version of the configuration cannot be edited to specifically change the address defined in the original CLNS HOST command. You must specifically change any commands that refer to the original address. This affects all commands that accept names.

The commands that are affected by these requirements include:

- NET (router subcommand)
- CLNS IS-NEIGHBOR (interface subcommand)
- CLNS ES-NEIGHBOR (interface subcommand)
- CLNS ROUTE (global configuration command)

In the following command example, the NET command specification assumes that a CLNS HOST command already has been entered (with the name Digital representing a valid NET).

```
router iso-igrp
net Digital
```

Static NSAP Address Assignment

The CLNS NET command can be used as either a global configuration command or as an interface subcommand to assign static addresses. The following brief descriptions explain these two uses.

Static NET Address for the Router

If a router is configured to support ISO CLNS, but is not configured to dynamically route CLNS packets using ISO-IGRP or IS-IS, use the CLNS NET command to assign an address to the router. The command syntax follows.

clns net *NET-address*
no clns net *NET-address*

There is no default for this address.

The argument *NET-address* is a NET.

A CLNP packet sent to any of the defined NSAPs or NETs will be received by the router. The router chooses the NET to use when it sends a packet with the following algorithm:

- If no dynamic routing protocol is running, use the NET defined for the outgoing interface if it exists; otherwise, use the NET defined for the router.
- If ISO-IGRP is running, use the NET of the routing process that is running on this interface.
- If IS-IS is running, use the NET of the IS-IS routing process that is running on this interface.

The NO CLNS NET command removes the NET or NSAP address.

Static NSAP Addresses for an Interface

The CLNS NET command can also be used as a configuration interface subcommand for assigning an NSAP address for a specific interface. The command syntax is as follows:

clns net {*NSAP-address* | *name*}
no clns net {*NSAP-address* | *name*}

The argument *NSAP-address* is an NSAP address; the argument *name* is a name to be associated with this interface. Either can be used.

The NO CLNS NET command removes the NSAP address. This command is useful if you are doing static routing and need to control the source NET used by the router on each interface.

Configuring DECnet Cluster Aliases

DECnet Phase V *cluster aliasing* allows multiple systems to advertise the same System ID in end-system Hello messages. The router does this by caching multiple ES adjacencies with the same NSAP, but different SNPA addresses. When a packet is destined to the common NSAP address, the router load-splits the packets among the different SNPA addresses. A router that supports this capability forwards traffic to each individual system.

The CLNS CLUSTER-ALIAS interface subcommand enables this on a per interface basis. The command syntax is as follows:

clns cluster-alias
no clns cluster-alias

If DECnet Phase V cluster aliases are disabled on an interface, End-System Hello packet information is used to replace any existing adjacency information for the NSAP. Otherwise, an additional adjacency (with a different SNPA) is created for the same NSAP.

Example

```
!
! Enable cluster aliasing for CLNS
!
clns nsap 47.0004.004d.0001.0000.0c00.1111.00
clns routing

interface Ethernet 0
clns cluster-alias

interface Serial 0
clns cluster-alias
```

Maintaining a CLNS Network

Use the EXEC commands described in this section to maintain the ISO CLNS caches, tables, and databases.

clear clns cache
The EXEC command CLEAR CLNS CACHE clears and reinitializes the CLNS routing cache.

clear clns route
The command CLEAR CLNS ROUTE removes all of the dynamically derived CLNS routing information.

clear clns neighbors
The command CLEAR CLNS NEIGHBORS removes CLNS neighbor information from the adjacency database.

clear clns es-neighbors
The command CLEAR CLNS ES-NEIGHBORS removes ES neighbor information from the adjacency database.

clear clns is-neighbors
The command CLEAR CLNS IS-NEIGHBORS removes IS neighbor information from the adjacency database.

Monitoring a CLNS Network

Use the EXEC commands described in this section to obtain displays of activity on the ISO CLNS network.

Displaying General CLNS Information

The SHOW CLNS command displays information about the CLNS network. Enter this command at the EXEC prompt:

show clns

Sample output follows.

```
Global CLNS Information:
  2 Interfaces Enabled for CLNS
  NET: 39.0004.0030.0000.0C00.224D.00
  NET: 39.0003.0020.0000.0C00.224D.00
  Configuration Timer: 60, Default Holding Timer: 300, Packet Lifetime 64
  ERPDU's requested on locally generated packets
  Intermediate system operation enabled (forwarding allowed)
  ISO-IGRP level-1 Router: remote
    Routing for Domain: 39.0003, Area: 0020
  ISO-IGRP level-2 Router: DOMAIN_remote
    Routing for Domain: 39.0003
  IS-IS level-1-2 Router:
    Routing for Area: 39.0004.0030
```

In the display:

- The first line indicates how many interfaces have the CLNS routing protocol turned on.
- The next several lines contain the NETs for this router. Note that there may be more than one NET for a router.
- The `Configuration Timer` field displays the frequency with which the router will send out IS Hello packets. The number following the `default holding timer` field is the length of time the router's Hello packets will be remembered. The `packet lifetime` displayed is the default value used in packets sourced by this router.
- The next line indicates whether error PDUs (ERPDUs) will be requested for packets sourced by the router.
- The next line of global information indicates whether or not this router is configured to be an End System or an Intermediate System. It is not generally useful to configure a router to be an End System.
- The subsequent fields specify what kinds of CLNS routing is enabled (ISO-IGRP or IS-IS) and the routing level (Level 1, Level 2, or both).

Displaying the CLNS Routing Cache

Use the EXEC command SHOW CLNS CACHE to display the CLNS routing cache. Enter this command at the EXEC prompt:

show clns cache

The cache contains an entry for each destination that has packet switching enabled.

Following is sample output:

```
CLNS routing cache version 433
Destination -> Next hop @ Interface : SNPA Address
*39.0004.0040.0000.0C00.2D55.00
  -> 0000.0C00.2D55 @ Serial1
 39.0003.0020.0000.0C00.3E51.00
  -> 0000.0C00.3E51 @ Ethernet0 : 0000.3000.5475
```

Note

A leading asterisk (*) in the first column indicates that the entry is valid.

In the display, there will be an entry in the cache specifying each destination for which the router has switched a packet recently, including the router itself. Each entry has the following format:

```
destination→  first hop address@interface:MAC-layer address
```

Displaying CLNS Traffic

Use the SHOW CLNS TRAFFIC command to list the CLNS packets this router has seen. Enter this command at the EXEC prompt:

show clns traffic

Sample output follows:

```
CLNS & ESIS Output: 139885, Input: 90406
CLNS Local: 0, Forward: 0
CLNS Discards:
  Hdr Syntax: 150, Checksum: 0, Lifetime: 0, Output cngstn: 0
  No Route: 0, Dst Unreachable 0, Encaps. Failed: 0
  NLP Unknown: 0, Not an IS: 0
CLNS Options: Packets 19, total 19, bad 0, GQOS 0, cngstn exprncd 0
CLNS Segments:  Segmented: 0, Failed: 0
CLNS Broadcasts: sent: 0, rcvd: 0
Echos: Rcvd 0 requests, 69679 replies
      Sent 69701 requests, 0 replies
ESIS(sent/rcvd): ESHs: 0/34, ISHs: 483/1839, RDs: 0/0, QCF: 0/0
ISO-IGRP: Querys (sent/rcvd): 0/0 Updates (sent/rcvd): 1279/1402
ISO-IGRP: Router Hellos: (sent/rcvd): 1673/1848
ISO-IGRP Syntax Errors: 0
IS-IS: Level-1 Hellos (sent/rcvd): 0/0
IS-IS: Level-2 Hellos (sent/rcvd): 0/0
IS-IS: PTP Hellos     (sent/rcvd): 0/0
IS-IS: Level-1 LSPs   (sent/rcvd): 0/0
IS-IS: Level-2 LSPs   (sent/rcvd): 0/0
IS-IS: Level-1 CSNPs  (sent/rcvd): 0/0
IS-IS: Level-2 CSNPs  (sent/rcvd): 0/0
IS-IS: Level-1 PSNPs  (sent/rcvd): 0/0
IS-IS: Level-2 PSNPs  (sent/rcvd): 0/0
IS-IS: Level-1 DR Elections: 0
IS-IS: Level-2 DR Elections: 0
IS-IS: Level-1 SPF Calculations: 0
IS-IS: Level-2 SPF Calculations: 0
```

In the display:

- The first line lists the total number of packets that this router has sent (the `Output` field) and received (the `Input` field).
- The `CLNS Local` field lists the number of packets that were generated by this router.
- The `CLNS Forward` field lists the number of packets that this router has forwarded.
- The `CLNS Discards` field lists the packets that CLNS has discarded, along with the reason for the discard.
- The `CLNS Options` field lists the options that have been seen in CLNS packets.
- The `CLNS Segments` field lists the number of packets that have been segmented and the number of failures that occurred because a packet could not be segmented.

- The `CLNS Broadcasts` field lists the number of CLNS broadcasts that have been sent and received.
- The `Echos` field lists the number of echo request packets and echo reply packets that have been received. The line following this field lists the number of echo request packets and echo reply packets that have been sent.
- The `ESIS(sent/rcvd)` field lists the number of ESH, ISH, and Redirects sent and received.
- The `ISO-IGRP` field lists the number of IGRP queries and updates sent and received.
- The `Router Hellos` field lists the number of IGRP router Hello packets that have been sent and received.
- The `IS-IS: Level-1 Hellos (sent/rcvd)` field lists the number of Level-1 IS-IS Hello packets sent and received.
- The `IS-IS: Level-2 Hellos (sent/rcvd)` field list the number of Level-2 IS-IS Hello packets sent and received.
- The `IS-IS: PTP Hellos (sent/rcvd)` field lists the number of point-to-point IS-IS Hello packets sent and received over serial links.
- The `IS-IS: Level-1 LSPs (sent/rcvd)` field lists the number Level-1 link state PDUs sent and received.
- The `IS-IS: Level-2 LSPs (sent/rcvd)` field lists the number Level-2 link state PDUs sent and received.
- The `IS-IS: Level-1 CSNPs (sent/rcvd)` field lists the number of Level-1 complete sequence number PDUs sent and received.
- The `IS-IS: Level-2 CSNPs(sent/rcvd)` field lists the number of Level-2 complete sequence number PDUs sent and received.
- The `IS-IS: Level-1 PSNPs (sent/rcvd)` field lists the number of Level-1 partial sequence number PDUs sent and received.
- The `IS-IS: Level-2 PSNPs (sent/rcvd)` field lists the number of Level-2 partial sequence number PDUs sent and received.
- The `IS-IS: Level-1 DR Elections` field lists the number of times Level-1 designated router election occurred.
- The `IS-IS: Level-2 DR Elections` field lists the number of times Level-2 designated router election occurred.
- The `IS-IS: Level-1 SPF Calculations` field lists the number of times Level-1 shortest-path-first tree was computed.
- The `IS-IS: Level-2 SPF Calculations` field lists the number of times Level-2 shortest path first tree was computed.

Displaying CLNS Redirect Information

The SHOW CLNS REDIRECTS command displays CLNS redirect information. Enter this command at the EXEC prompt:

show clns redirects

Only ESes maintain redirect information.

Displaying Status About Specific Interfaces

The SHOW CLNS INTERFACE command lists the CLNS-specific information about each interface and is entered at the EXEC prompt, as follows:

show clns interface [*interface unit*]

Following is sample output illustrates information displayed for Serial interfaces:

```
Serial 1 is up, line protocol is up
  Checksums enabled, MTU 1497, Encapsulation HDLC
ERPDUs enabled, min. interval 10 msec.
  RDPDUs enabled, min. interval 100 msec., Addr Mask enabled
  Congestion Experienced bit set at 4 packets
  CLNS fast switching enabled
  DEC compatibility mode OFF for this interface
  Next ESH/ISH in 48 seconds
  Routing Protocol: IS-IS
    Circuit Type: level-1-2
    Level-1 Metric: 10, Priority: 64, Circuit ID: 0000.0C00.2D55.0A
    Level-2 Metric: 10, Priority: 64, Circuit ID: 0000.0000.0000.00
```

In the display:

- The `Checksums enabled` field can be either enabled or disabled. The number following `MTU` is the maximum transmission size for a packet on this interface. The `Encapsulation` field describes the encapsulation used by CLNP packets on this interface.
- The next line displays information about the generation of error PDUs (`ERPDUs`). They can be either enabled or disabled. If they are enabled, they will be sent out no more frequently than the specified interval.
- The next line provides information about the generation of redirect PDUs (`RDPDUs`). They can be either enabled or disabled. If they are enabled, they will be sent out no more frequently than the specified interval. If the address mask is enabled, redirects will be sent out with an address mask.
- The next line tells when CLNS will turn on the congestion experienced bit. The default is to turn this bit on when there are more than four packets in a queue.
- The next line displays whether or not fast switching is supported for CLNS on this interface.
- The next line indicates whether DEC compatibility has been enabled.
- The `Next` field displays when the next ESH or ISH will be sent on this interface.
- The next line lists the areas that this interface is in. In most cases, an interface will be in only one area.
- The last series of fields displays information pertaining to the ISO CLNS routing protocols enabled on the interface. For ISO-IGRP, the routing domain and area addresses are specified. For IS-IS, the Level 1 and Level 2 metrics, priorities, and Circuit IDs are specified.

Displaying ES and IS Neighbors

The SHOW CLNS NEIGHBORS command displays both ES and IS neighbors. Enter this command at the EXEC prompt:

show clns neighbors [*interface-type unit*] [**detail**]

The *interface-type* and *unit* optional arguments specify a specific interface to be displayed. If **detail** is specified, the area addresses advertised by the neighbor in the Hello messages is displayed. Otherwise, a summary display is provided. This display is a composite of the SHOW CLNS ES-NEIGHBOR and SHOW CLNS IS-NEIGHBOR commands.

Sample output follows:

```
System Id       SNPA             Interface   State  Holdtime  Type  Protocol
0000.0000.0007  aa00.0400.6408   Ethernet0   Init   277       IS    ES-IS
0000.0C00.0C35  0000.0c00.0c36   Ethernet0   Up     91        L1    IS-IS
0800.2B16.24EA  aa00.0400.2d05   Ethernet0   Up     29        L1L2  IS-IS
0800.2B14.060E  aa00.0400.9205   Ethernet0   Up     1698      ES    ES-IS
0000.0C00.3E51  *HDLC*           Serial1     Up     28        L2    IS-IS
0000.0C00.62E6  0000.0c00.62e7   Ethernet0   Up     22        L1    IS-IS
AA00.0400.2D05  aa00.0400.2d05   Ethernet0   Init   24        IS    ES-IS
```

The following fields are provided in this display:

- `System ID`—The six-byte value that identifies a system in an area. A name is displayed in this field if one has been assigned with the clns host command.
- `SNPA`—Subnetwork Point of Attachment. This is the data link address.
- `Interface`—The interface in which the system was learned from.
- `State`— The state of the ES or IS. Possible values are as follows:

  `Init`—The system is an IS and is waiting for an IS-IS Hello message. IS-IS regards the neighbor as not adjacent.

  `Up`—Believes the ES or IS is reachable.
- `Holdtime`—The number of seconds the neighbor indicates that it is valid.
- `Type`—The adjacency type. Possible values are as follows:

  `ES`—An end-system adjacency either discovered via the ES-IS protocol or statically configured.

  `IS`—A router adjacency either discovered via the ES-IS protocol or statically configured.

  `L1`—A router adjacency for Level-1 routing only.

  `L1L2`—A router adjacency for Level-1 and Level-2 routing.

  `L2`—A router adjacency for Level-2 only.

- Protocol—The protocol through which the adjacency was learned. Valid protocol sources are ES-IS, IS-IS, ISO-IGRP, and Static.

A sample output of the SHOW CLNS NEIGHBORS DETAIL command follows:

```
System Id       SNPA             Interface   State  Holdtime  Type  Protocol
0000.0000.0007  aa00.0400.6408   Ethernet0   Init   291       IS    ES-IS
  Area Address(es): 47.0005.80FF.F500.0000.0003.0020
0000.0C00.0C35  0000.0c00.0c36   Ethernet0   Up     94        L1    IS-IS
  Area Address(es): 47.0004.004D.0001 39.0001
0800.2B16.24EA  aa00.0400.2d05   Ethernet0   Up     9         L1L2  IS-IS
  Area Address(es): 47.0004.004D.0001
0800.2B14.060E  aa00.0400.9205   Ethernet0   Up     1651      ES    ES-IS
  Area Address(es): 49.0040
0000.0C00.3E51  *HDLC*           Serial1     Up     27        L2    IS-IS
  Area Address(es): 39.0004
0000.0C00.62E6  0000.0c00.62e7   Ethernet0   Up     26        L1    IS-IS
  Area Address(es): 47.0004.004D.0001
AA00.0400.2D05  aa00.0400.2d05   Ethernet0   Init   29        IS    ES-IS
  Area Address(es): 47.0004.004D.0001
```

In addition to the information displayed in SHOW CLNS NEIGHBORS, this SHOW command displays the area addresses associated with the adjacencies.

Displaying CLNS IS Neighbors

The SHOW CLNS IS-NEIGHBORS command displays neighbor entries sorted according to the area in which they are located. Enter this command at the EXEC prompt:

show clns is-neighbors [*interface-type unit*] [**detail**]

The *interface-type* and *unit* optional arguments specify an interface to be displayed. If **detail** is specified, the areas associated with intermediate systems are displayed. Otherwise, a summary display is provided. Following is sample output:

```
System Id       Interface   State  Type  Priority  Circuit Id         Format
0000.0C00.0C35  Ethernet0   Up     L1    64        0000.0C00.62E6.03  Phase V
0800.2B16.24EA  Ethernet0   Up     L1L2  64/64     0800.2B16.24EA.01  Phase V
0000.0C00.3E51  Serial1     Up     L2    0         04                 Phase V
0000.0C00.62E6  Ethernet0   Up     L1    64        0000.0C00.62E6.03  Phase V
```

This display presents IS-IS related information for IS-IS router adjacencies. The following fields are provided in this display:

- System ID—The identification value of the system. A name is displayed in this field if one has been assigned with the clns host command.
- Interface—The interface on which the router was discovered.
- State—The adjacency state. Up and Init are the states. See the SHOW CLNS NEIGHBORS description.
- Type—L1, L2, andL1L2 type adjacencies. See the SHOW CLNS NEIGHBORS description.
- Priority—The router priority that the respective neighbor is advertising. The highest priority neighbor is elected the designated router for the interface.

- `Circuit ID`—The neighbor's idea of what is the designated router for the interface. A name is displayed in this field if one has been assigned with the CLNS HOST command.
- `Format`—If the neighbor is a `Phase V` (OSI) adjacency or `Phase IV` (DECnet) adjacency.

A sample output of the SHOW CLNS IS-NEIGHBORS DETAIL command follows:

```
System Id       Interface   State  Type  Priority  Circuit Id        Format
0000.0C00.0C35  Ethernet0   Up     L1    64        0000.0C00.62E6.03  Phase V
  Area Address(es): 47.0004.004D.0001 39.0001
0800.2B16.24EA  Ethernet0   Up     L1L2  64/64     0800.2B16.24EA.01  Phase V
  Area Address(es): 47.0004.004D.0001
0000.0C00.3E51  Serial1     Up     L2    0         04                 Phase V
  Area Address(es): 39.0004
0000.0C00.62E6  Ethernet0   Up     L1    64        0000.0C00.62E6.03  Phase V
  Area Address(es): 47.0004.004D.0001
```

In addition to the information displayed in SHOW CLNS IS-NEIGHBORS, this SHOW command displays the area addresses associated with the adjacencies.

Displaying CLNS ES Neighbors

The SHOW CLNS ES-NEIGHBORS command lists the ES neighbors that this router knows about. Enter this command at the EXEC prompt:

show clns es-neighbors [*interface-type unit*] [**detail**]

The *interface-type* and *unit* optional arguments specify an interface to be displayed. If **detail** is specified, the areas associated with the end systems are displayed. Otherwise, a summary display is provided. Following is sample output:

```
System Id       Interface   State  Type  Format
0800.2B14.060E  Ethernet0   Up     ES    Phase V
0800.2B14.0528  Ethernet0   Up     ES    Phase V
```

This displays end system adjacencies. See the SHOW CLNS IS-NEIGHBORS command for field descriptions.

A sample output of the SHOW CLNS ES-NEIGHBORS DETAIL command follows:

```
System Id       Interface   State  Type  Format
0800.2B14.060E  Ethernet0   Up     ES    Phase V
  Area Address(es): 49.0040
0800.2B14.0528  Ethernet0   Up     ES    Phase V
  Area Address(es): 49.0040
```

In addition to the information displayed in SHOW CLNS ES-NEIGHBORS, this SHOW command displays the area addresses associated with the adjacencies.

ISO CLNS PING Command

The OSI Connectionless Network Protocol (ISO 8473) does not specify a network-level echo protocol. The Internet Engineering Task Force (IETF) has specified and proposed such a protocol in RFC 1139. The DECbrouter 90 implements this specification using the proposed new PDU types Echo Request (1E) and Echo Reply (1F). Other manufacturers' routers may or may not forward these packets, depending on whether they are specific about the packet types they will forward. End systems may not recognize these packets, but will typically generate an error packet (ERPDU) as a response. This ERPDU is useful, as it confirms the reachability of the end system. Table 1–1 lists the characters displayed during the **ping** test and their meanings.

Table 1–1 Ping Test Characters

Char	Meaning
!	Each exclamation point indicates receipt of a reply.
.	Each period indicates the network server timed out while waiting for a reply.
U	A destination unreachable error PDU was received.
C	A congestion experienced packet was received.
I	User interrupted test.
?	Unknown packet type.
&	Packet lifetime exceeded.

The output concludes with the success rate and minimum, average, and maximum round-trip times. To abort a **ping** session, type the escape sequence (by default, type Ctrl/^, X, which is done by simultaneously pressing the Ctrl, Shift, and 6 keys, letting go, then pressing the x key).

Example 1

The following example uses a name to specify the source:

```
Protocol [ip]: clns
Target CLNS address: thoth
Repeat count [5]:
Datagram size [100]:
Timeout in seconds [2]:
Type escape sequence to abort.
Sending 5, 100-byte CLNS Echos
        to 55.0006.0100.0000.0000.0001.8888.1112.1314.151
6.00, timeout is 2 seconds:
!!!!!
Success rate is 100 percent, round-trip min/avg/max = 112/113/116 ms
```

Example 2

In this example, an NET address is specified.

```
Protocol [ip]: clns
Target CLNS address: 47.0004.0050.0002.0000.0c00.243b.00
Repeat count [5]:
Datagram size [100]:
Timeout in seconds [2]:
Type escape sequence to abort.
Sending 5, 100-byte CLNS Echos to 47.0004.0050.0002.0000.0C00.243B.00,
        timeout is 2 seconds:
!!!!!
Success rate is 100 percent, round-trip min/avg/max = 1/4/8 ms
```

ISO CLNS TRACE Command

The ISO CLNS TRACE command allows you to discover the path that packets take through your network. It sends ping packets and takes advantage of the ERPDUs that are generated when a packet exceeds its time-to-live (TTL) value. The TRACE command offers default and settable parameters for specifying a simple or extended trace mode. Enter the following command at the EXEC prompt:

trace [*destination*]

To invoke a simple trace test, enter the destination address or host name on the command line. The default parameters for the appropriate protocol are assumed and the tracing action begins.

To use nondefault parameters and invoke an extended TRACE test, enter the command without a destination argument. You will be stepped through a dialog to select the desired parameters, listed in Table 15-2.

Typing the escape sequence (default, type Ctrl/^, X, which is done by simultaneously pressing the Ctrl, Shift, and 6 keys, letting go then pressing the x key) terminates a TRACE command.

How Trace Works

The TRACE command works by taking advantage of the error messages generated by routers when a datagram exceeds its time-to-live (TTL) value.

The TRACE command starts by sending probe datagrams with a TTL value of one. This causes the first router to discard the probe datagram and send back an error message. The trace command sends several probes at each TTL level and displays the round trip time for each.

The TRACE command sends out one probe at a time. Each outgoing packet may result in one of two error messages. A *time exceeded* error message indicates that an intermediate router has seen and discarded the probe. A *destination unreachable* error message indicates that the destination node has received the probe and discarded it because it could not deliver the packet. If the timer goes off before a response comes in, TRACE prints an asterisk (*).

The TRACE command terminates when the destination responds, when the maximum TTL was exceeded, or when the user interrupts the trace with the escape sequence. The information is encoded as follows:

```
hop-count name(nsap) result-of-probe
```

Tracing CLNS Routes

You can use the TRACE command to trace routes on a router configured with the ISO CLNS protocol. When stepping through the TRACE dialog for CLNS, the parameters that follow can be specified.

- `Protocol [ip]`. The default protocol for TRACE is IP. You must specify CLNS to begin tracing a router on a CLNS router.
- `Target CLNS address`. You can specify either an NSAP or host name.
- `Timeout in seconds`. You can specify the length of time to wait after sending each probe before giving up on getting a response.
- `Probe count`. You can specify the number of probes to be sent at each TTL level. The default is three.
- `Minimum Time to Live [1]`: You can set the TTL value for the first probes. The default is 1. Set to a higher value to suppress the display of known hops.
- `Maximum Time to Live [30]`: You can set the largest TTL value that can be used. The default is 30. The TRACE command terminates when the destination is reached or when this value is reached.

Table 1–2 describes the output.

Table 1–2 Trace Test Characters

Char	Meaning
nn msec	For each node, the round-trip time (in *nn* milliseconds) for the specified number of probes.
&	A time-to-live-exceeded error PDU was received.
U	A destination unreachable error PDU was received.
I	The user interrupted the test.
*	The probe timed out.
C	A congestion experienced packet was received.

Following is simple **trace** output:

```
Protocol [ip]: clns
Target CLNS address: thoth
Timeout in seconds [3]:
Probe count [3]:
Minimum Time to Live [1]:
Maximum Time to Live [30]:
Type escape sequence to abort.
Tracing the route to THOTH (55.0006.0100.0000.0000.0001.8888.1112.1314.1516)
  1 HORUS(55.0006.0100.0000.0000.0001.6666.3132.3334.3536)
        32 msec ! 28 msec ! 28 msec !
  2 ISIS(55.0006.0100.0000.0000.0001.7777.2122.2324.2526)
        56 msec ! 80 msec ! 56 msec !
  3 THOTH(55.0006.0100.0000.0000.0001.8888.1112.1314.1516)
        80 msec ! 80 msec ! 8
```

This example traces a more complex route:

```
Protocol [ip]: clns
Target CLNS address: cerdiwen
Timeout in seconds [3]:
Probe count [3]:
Minimum Time to Live [1]:
Maximum Time to Live [30]:
Type escape sequence to abort.
Tracing the route to CERDIWEN (49.BEAD.0000.0C00.40A7.00)
  1 DEMETER(49.BEAD.0000.0C00.40AF.00) 72 msec !
     ARTEMIS(49.BEAD.0000.0C00.2D57.00) 72 msec !
     DEMETER(49.BEAD.0000.0C00.40AF.00) 76 msec !
  2 CERDIWEN(49.BEAD.0000.0C00.40A7.00) 148 msec ! 148 msec ! 144 msec !
```

The output from the TRACE command displays information for each probe that it sends out. As you can see, there were two equal cost paths to the destination. The first packet went via Demeter, the second went via Artemis, and the third went via Demeter again.

Debugging a CLNS Network

Use the EXEC commands described in this section to troubleshoot and monitor the ISO CLNS network transactions. For each DEBUG command, there is a corresponding UNDEBUG command that turns the message logging off. Generally, you will enter these commands during troubleshooting sessions with Digital engineers.

debug clns-esis-events

The DEBUG CLNS-ESIS-EVENTS command traces the more unusual ES-IS events, including previously unknown neighbors, neighbors that have aged out, and neighbors that have changed roles (ES to IS, and so on).

debug clns-esis-packets

The DEBUG CLNS-ESIS-PACKETS command traces ES-IS activity, including sending and receiving ESHs and ISHs, receiving Redirects (RDs), and aging out ESH/ISH/RD entries.

debug clns-events

The DEBUG CLNS-EVENTS command traces the more unusual CLNS events, including packet discards, sending redirects, and so forth.

debug clns-packets

The DEBUG CLNS-PACKETS command causes all CLNS activity to be traced, including packet forwarding. You can use this to circumvent a potential problem with the PING command in mixed Digital and non-Digital installations; see ISO CLNS PING Command for more information.

debug clns-routing

The DEBUG CLNS-ROUTING command causes all CLNS routing table activity to be traced.

ISO CLNS Global Configuration Command Summary (Switching)

This section provides an alphabetical summary of the ISO CLNS global configuration commands.

[**no**] **clns configuration-time** *seconds*

Specifies the rate at which ESHs and ISHs are sent. The default value for *seconds* is 60.

[**no**] **clns holding-time** *seconds*

Allows the sender of an ESH or ISH to specify the length of time during which the information in the Hello packets will be believed. The argument *seconds* specifies the time in seconds. The default value is 300 seconds (five minutes).

clns host *name nsap*

Defines a name for an NSAP that can then be used in commands requiring NSAPs.

[**no**] **clns net** *NET-address*

Used as a global configuration command to assign a static address for a router. If a router is configured to support ISO CLNS but is not configured to dynamically route CLNS packets using a ISO-IGRP or IS-IS, use this command to assign an address to the router. There is no default for this address. The NO CLNS NET command removes any previously configured NET-address.

[**no**] **clns packet-lifetime** *number*

Specifies the initial lifetime for locally generated packets. The default value for *number* is 64.

[**no**] **clns routing**

Enables or disables routing of CLNS packets.

ISO CLNS Interface Subcommand Summary (Switching)

This section provides an alphabetical list of the ISO CLNS interface subcommands.

[**no**] **clns checksum**

Enables or disables checksum generation when ISO CLNS routing software sources a CLNS packet. By default, this function is on. Use the **no** form of the command to disable checksum generation.

[**no**] **clns cluster-alias**

Allows multiple systems to advertise the same system ID in end-system Hello messages by caching multiple ES adjacencies with the same NSAP, but different SNPA addresses. When a packet is destined to the common NSAP address, the router load-splits the packets among the different SNPA addresses. A router that supports this capability forwards traffic to each individual system.

[**no**] **clns congestion-threshold** *number*

Sets the congestion experience bit if the output queue has more than the specified number of packets in it. A *number* value of zero or the **no** form of the command prevents this bit from being set. The default value for *number* is four.

[**no**] **clns enable**

If you do not intend to perform any static or dynamic routing on an interface, but intend to pass ISO CLNS packet traffic to end systems, use the CLNS ENABLE interface subcommand.

[**no**] **clns erpdu-interval** *milliseconds*

Determines the minimum interval time, in milliseconds, between ERPDUs. A *milliseconds* value of zero or the **no** form of the command turns off the interval rate and effectively sets no limit to the ERPDU rate. The default rate is once every ten milliseconds.

[**no**] **clns es-neighbor** *nsap snpa*

Lists all end systems that will be used when mapping information is statically entered. The SNPAs are the MAC addresses.

[**no**] **clns es-neighbor** *nsap snpa* [*X.25-facilities-info*]

Lists all end systems that will be used when mapping information is statically entered. The SNPAs are the X.25 network addresses (X.121 addresses). These are usually assigned by the X.25 network provider. Use the argument X.25-facilities-info to specify nondefault packet and window size, reverse charge information, and so on.

[**no**] **clns is-neighbor** *nsap snpa*

Lists all intermediate systems that will be used when mapping information is statically entered. The SNPAs are the MAC addresses.

[**no**] **clns is-neighbor** *nsap snpa* [*X.25-facilities-info*]

Lists all intermediate systems that will be used when mapping information is statically entered. The SNPAs are the X.25 network addresses (X.121 addresses). These are usually assigned by the X.25 network provider. Use the argument *X.25-facilities-info* to specify nondefault packet and window size, reverse charge information, and so on.

[**no**] **clns mtu** *size*

Sets the MTU packet size for the interface. The minimum value for *size* is 512. The **no** form of the command restores the default and maximum packet size.

[**no**] **clns net** {*NSAP-address* | *name*}

Used as an interface subcommand to assign an NSAP address or name to a router interface. If a router is configured to support ISO CLNS, but is not configured to dynamically route CLNS packets using a ISO-IGRP or IS-IS, use this command to assign an address to the router. There is no default for this address. The NO CLNS NET command removes any previously configured NSAP-address.

[**no**] **clns rdpdu-interval** *milliseconds*

Determines the minimum interval time, in milliseconds, between RDPDUs. A *milliseconds* value of zero or the **no** keyword turns off the interval rate and effectively sets no limit to the RDPDU rate. The default rate is once every 100 milliseconds.

[**no**] **clns rdpdu-mask**

Enables or disables the address mask on RDPDUs. The address mask is normally present on all RDPDUs, but can be disabled with the NO CLNS RDPDU-MASK command.

Note

SNPA masks are never sent, and the router ignores RDPDUs when it is acting as an IS.

[**no**] **clns route-cache**

Allows fast switching through the cache, and by default, is enabled. To disable fast switching, use the **no** keyword.

Note

The cache still exists and is used after the NO CLNS ROUTE-CACHE command is used; the software just does not do fast switching through the cache.

[**no**] **clns send-erpdu**

Allows or prevents CLNS to send an error PDU when it detects an error in a data PDU, and by default, is enabled. To disable this function, use the **no** keyword.

[**no**] **clns send-rdpdu**

Allows or prevents CLNS to send redirect PDUs when a better route for a given host is known. This is the default behavior. To disable this function, use the **no** keyword.

[**no**] **clns want-erpdu**

Specifies whether to request error PDUs on packets sourced by the router. The default is to request error PDUs.

2 Configuring ISO CLNS Routing Protocols

This chapter describes routing protocols supported by the International Organization for Standardization (ISO) Connectionless Network Services (CLNS) protocol. Chapter 1 of the *DECbrouter 90 Products Configuration and Reference, Volume 3* publication contains all the information you need for configuring ISO CLNS switching, as well as configuration requirements not related specifically to ISO CLNS routing protocols support. This chapter focuses on ISO CLNS routing protocols. Topics in this chapter include:

- An introduction to supported ISO CLNS routing protocols
- Overview of the sequence for configuring and starting routing processes
- Details of commands used for configuring static and dynamic routing
- Examples of configuring the supported ISO CLNS protocols˙static, ISO-IGRP, and IS-IS routing
- Information concerning monitoring and debugging ISO CLNS routing processes on your routers
- Summaries of ISO CLNS routing configuration commands

This chapter also explains how to optimize and fine-tune your CLNS-based internetwork. Configuration examples are grouped in a separate section in this chapter, with configuration command lists and illustrations.

Implementation of ISO CLNS

The DECbrouter 90T routing software supports packet forwarding and routing for ISO CLNS on networks using a variety of network interfaces including Ethernet and serial.

You can use the CLNS protocol on serial interfaces with HDLC, LAPB, X.25, SMDS, PPP, or Frame Relay encapsulation. To use HDLC encapsulation, you must have a DECbrouter 90T or Cisco Systems router at each end of the link. If you use X.25 encapsulation, you must manually enter the NSAP-to-X.121 mapping. The LAPB, SMDS, Frame Relay, and X.25 encapsulations will interoperate with other vendors' devices.

This CLNS implementation is compliant with the Government Open Systems Interconnection Profile (GOSIP) Version 2.

The following ISO and ANSI standards are fully supported:

- ISO 9542— Documents the End System-to-Intermediate System (ES-IS) routing exchange protocol
- ISO 8473— Documents the ISO Connectionless Network Protocol (CLNP)
- ISO 8348/Ad2— Documents Network Service Access Points (NSAPs)

- ISO 10589— Documents Intermediate System-to-Intermediate System (IS-IS) Intradomain Routing Exchange Protocol

ISO CLNS supports static routes and dynamic routing of IS-IS and ISO-IGRP routing protocols.

DECbrouter 90T-Supported ISO Routing Protocols

The basic function of a router is to forward packets to the proper destination. All routers do this by looking up the destination address in a routing table. Routing tables can be built either dynamically or statically. If you configure all of the entries in the table, you are using *static* routing. If a routing process builds the tables, you are using *dynamic* routing. It is possible, and sometimes necessary, to use both static and dynamic routing simultaneously.

The DECbrouter 90T supports two dynamic routing protocols for CLNP networks: *ISO-IGRP* and *IS-IS*. ISO-IGRP was developed by Cisco Systems. Both routing protocols support the concept of *areas*. Within an area, all routers can reach all of the Station IDs. Between areas, routers can reach the proper area.

Support for IS-IS on TCP/IP networks is documented in *DECbrouter 90 Products Configuration and Reference, Volume 1*.

IS-IS supports two levels of routing: *station routing* (within an area) and *area routing* (between areas).

ISO CLNS Routing Configuration Overview

Follow these steps to configure your router for ISO CLNS routing:

STEP 1:	Enable CLNS routing with the CLNS ROUTING global configuration command (described in Chapter 1 of the *DECbrouter 90 Products Configuration and Reference, Volume 3* publication).
STEP 2:	Create dynamic routing processes with the **router** global configuration command and the **isis** or **iso-igrp** keywords. You do not need to explicitly specify a routing process to use static routing facilities.
STEP 3:	Use the CLNS ROUTER interface subcommand with each routing process defined in step 2 to specify the interface on which you want to route ISO CLNS. If you do not intend to use a dynamic routing protocol on a specific interface but want to switch packets over the link, use the CLNS ENABLE interface subcommand (described in Chapter 1 of the *DECbrouter 90 Products Configuration and Reference, Volume 3* publication).

Steps 1 through 3 enable CLNS routing. Optional commands are also available for customizing the routing environment, and you may need to follow steps 4 through 6 to complete your configuration.

STEP 4:	Redistribute routing information.
STEP 5:	Map NSAP addresses to media addresses.
STEP 6:	Adjust ES-IS parameters as necessary (described in Chapter 1 of the *DECbrouter 90T Configuration and Reference Volume 3* publication).

Configuring CLNS Static Routing

Static routing is used when it is not possible or desirable to use dynamic routing. For example, if you are using routers that do not support the same dynamic routing protocol, you must use static routing. In addition, if your network includes WAN links that require payment for connect time or have a cost per packet, you might not want to run a routing protocol over that link. Finally, if you want routers to advertise connectivity to external networks, but you are not running an interdomain routing protocol, you must use static routes.

Note

Only static routes that are configured with a *next-hop-NET* that changes interfaces can reroute around static links.

Static routes are entered by specifying *NSAP-prefix*/*next-hop-NET* pairs. Network Service Access Point (NSAP) and Network Entity Title (NET) describe ISO network addresses. NET is similar in function to an NSAP. In the routing table, the best match is the longest NSAP-prefix entry that matches the beginning of the destination NSAP.

Configuring a Static Route for a Router

Use the following global configuration command to configure a static route for a router:

clns route *nsap-prefix* {*next-hop-NET* | *name*}
no clns route *nsap-prefix*

The argument *nsap-prefix* is the front portion of an NSAP address. The argument *next-hop-NET* is the NET address of the node to which packets, or Protocol Data Units (PDUs), will be sent next. The argument name is the name of the node to which PDUs will be sent next. NSAPs that start with the value given for the argument *nsap-prefix* are forwarded to *next-hop-NET* or the next hop node *name*.

Examples

In the following sample static routing table (Table 2–1), the next-hop NETs are listed for completeness, but the next-hop NETs are not necessary for understanding the routing algorithm. Table 2–2 offers examples of how the routing entries in Table 2–1 can be applied to various NSAPs.

Table 2–1 Sample Routing Table Entries

Entry #1	NSAP Prefix	Next Hop NET
1	47.0005.000c.0001	47.0005.000c.0001.0000.1234.00
2	47.0004	47.0005.000c.0002.0000.0231.00
3	47.0005.0003	47.0005.000c.0001.0000.1234.00
4	47.0005.000c	47.0005.000c.0004.0000.0011.00
5	47.0005	47.0005.000c.0002.0000.0231.00

Table 2–2 Hierarchical Routing Examples

Datagram Destination NSAP	Table Entry Number Used
47.0005.000c.0001.0000.3456.01	1
47.0005.000c.0001.6789.2345.01	1
47.0004.1234.1234.1234.1234.01	2
47.0005.0003.4321.4321.4321.01	3
47.0005.000c.0004.5678.5678.01	4
47.0005.000c.0005.3456.3456.01	5
47.0005.0023.9876.9876.9876.01	6

Octet boundaries must be used for the internal boundaries of NSAPs and NETs.

For example, the routing table contains the following NSAP prefix entries:

1. 39.0001
2. 39.0002
3. 39.0001.0002

Data packets with the following destination NSAPs will match the routing table entries as follows:

- 39.0001.0002.0000.0c00.1111.00 uses route 3
- 39.0001.0003.0000.0c00.1111.00 uses route 1
- 39.0002.0002.0000.0c00.1111.00 uses route 2

The first address matches prefix-entry 3 instead of prefix-entry 1 because 3 has a longer match (39.0001.0002) than 1(39.0001).

The following example uses the global version of the CLNS route command to set a static route:

```
clns route 47.0004.000c  47.0005.0001.0000.0001.0000.00
```

Configuring Static Routes for an Interface

Use the following interface configuration command to configure a static route for an interface, as follows:

clns route *nsap-prefix interface-type unit*[*SNPA-address*]
no clns route *nsap-prefix*

The argument *nsap-prefix* is the front portion of a Network Service Access Point (NSAP) address. The argument *interface-type* is the type of interface for which you want configure a static route. The argument *unit* is the interface unit number.

The optional argument *SNPA-address* is a SubNetwork Point of Attachment (SNPA) data link address (for example, an Ethernet, X.25, or Frame Relay address) and is required for multiaccess networks that provide multiple neighbors on the other side of the link.

The CLNS ROUTE command is not literally applied to a specific interface. There are two forms of the CLNS ROUTE command, as follows:

- Configured with the destination NSAP and next-hop NET (described in the Configuring a Static Route for a Router).
- Configured with the destination NSAP, next-hop interface, and SNPA (described in this section). This is referred to as an interface-static route.

Example

This example sets static routes for specific interfaces serial 0 and Ethernet 0:

```
clns route 39.0001 serial 0
clns route 38.0002 ethernet0 0000.0c00.1550
!SNPA address specified
clns route 39.0003 serial 1 4085551021
!X.121 address specified
```

Discarding Packets

The following variation of the clns route global configuration command uses the discard keyword. Use the following global command to explicitly tell a router to discard packets that have the specified *nsap-prefix*:

clns route *nsap-prefix* **discard**
no clns route *nsap-prefix*

Example

In the following example, packets that have the NSAP prefix 39.0001 will be discarded.

```
clns route 39.0001 discard
```

Configuring ISO-IGRP CLNS Dynamic Routing

Use the commands described in this section to configure CLNS dynamic routing. Chapter 1 provides information on how to enable CLNS processing. After CLNS is enabled, to specify an ISO-IGRP routing process, you must enable the ISO-IGRP routing process and identify the address for the router.

Use the following global configuration command to specify an ISO-IGRP routing process:

router iso-igrp [*tag*]
no router iso-igrp [*tag*]

The keyword **router iso-igrp** specifies dynamic routing using the ISO-IGRP protocol. You can specify up to ten ISO-IGRP processes. The argument *tag* defines a meaningful name for routing process. For example, you could define a routing process named *Finance* for the Finance department, and another routing process named *Marketing* for the Marketing department. If defined here, you will use the *tag* names when configuring routing.

Note

The argument *tag* is optional; if it is not specified a null tag is assumed. If used, the *tag* argument must be unique among all CLNS router processes for a given router.

The NO ROUTER ISO-IGRP global configuration command with the appropriate tag disables ISO-IGRP routing for the system.

Example

In the following example, a router is specified in *Manufacturing*. (The command must be typed on one line.)

```
router iso-igrp Manufacturing
```

Specifying Router Level Support

If level-1 is specified in a CLNS command, the router acts as a station router. If level-1-2 is specified, the router acts as both a station router and an area router. If level-2 is specified, the router acts as an area router only.

Use the following router subcommand to enable the routing process on each interface supported by ISO-IGRP:

clns router iso-igrp *tag* [**level 2**]
no clns router iso-igrp *tag*

The argument *tag* is the tag defined for the routing process in the preceding router ROUTER ISO-IGRP global configuration command.

If you want this interface to advertise Level 2 information only, use the **level 2** keyword. This option reduces the amount of router-to-router traffic by telling the router to send out only Level 2 routing updates on certain interfaces. The Level 1 information will not be passed on the interfaces for which the Level 2 option is set.

Use the NO CLNS ROUTER ISO-IGRP command with the appropriate tag to disable the CLNS routing protocol on the interface.

Example

In the following example, the interface serial 0 will advertise only Level 2 information:

```
interface serial 0
clns router iso-igrp marketing level 2
```

Importing Routes Learned by other CLNS Routing Protocols

A router can be configured to perform interdomain dynamic routing by putting it into two domains and configuring it to redistribute the routing information between the domains. Routers configured this way are referred to as *border* routers.

Note

It is not necessary to use redistribution between areas.

If you have a router that is in two routing domains, you may want to redistribute routing information between the two domains. The REDISTRIBUTE router configuration subcommand configures routes redistributed into the ISO-IGRP domain.

Use the following router subcommand to configure which routes are redistributed into the ISO-IGRP domain:

redistribute *CLNS-routing-protocol tag*
no redistribute *CLNS-routing-protocol tag*

The argument *CLNS-routing-protocol tag* specifies the source of routes that will be redistributed into ISO-IGRP. The *tag* relates to the routing process that has been specified with the ROUTER ISO-IGRP global configuration subcommand discussed earlier in this chapter.

The supported *CLNS-routing-protocol* keywords are **isis**, and **static**. Static routes are only redistributed into ISO-IGRP when a REDISTRIBUTE STATIC command (described later in this chapter) is entered. The default is to not redistribute static routes into ISO-IGRP.

Example

The following example illustrates redistribution of IS-IS routes of router France and ISO-IGRP routes of router Germany into the ISO-IGRP area tagged Backbone:

```
router iso-igrp Backbone
redistribute isis France
redistribute iso-igrp Germany
```

Redistributing Static Routes

Use the following router subcommand to specify the routing process that will advertise static CLNS prefix routes:

redistribute static [**clns**]
no redistribute static [**clns**]

Use the NO REDISTRIBUTE STATIC command to stop the routing process from advertising static routes. Default is **redistribute static**.

Example

In the following example, the router will advertise any static routes in the Chicago domain about which it has been informed.

```
router iso-igrp Chicago
redistribute static
```

Note

Only the router that injects the static route requires a REDISTRIBUTE configuration command.

Specifying Preferred Routes

Use the following router subcommand to configure the administrative distance for CLNS routes learned:

distance *value* [**clns**]
no distance *value* [**clns**]

The argument *value*is the administrative distance, indicating the trustworthiness of a routing information source. This argument has a numeric value between 0 and 255. A higher relative value indicates a lower trustworthiness rating. Preference is given to routes with smaller values. The default values are as follows:

- Static routes—10
- ISO-IGRP routes—100
- IS-IS routes—110

The keyword **clns** is optional. Default is **clns**.

Example

In the following example, the distance value for CLNS routes learned is 90. Preference is given to these CLNS routes over routes with the default administrative distance value of 110.

```
router isis
distance 90 clns
```

Enabling and Disabling Split Horizon

Split horizon blocks information about routes from being advertised out the interface from which that information originated. This features usually optimizes communication among multiple routers, particularly when links are broken.

Use the following router subcommand to implement split horizons for ISO-IGRP updates:

clns split-horizon
no clns split-horizon

The default value for all LAN interfaces is **split-horizon**; the default value for WAN interfaces on X.25, Frame Relay, or SMDS networks is **no split-horizon**.

For more information about split horizon, see the *DECbrouter 90 Products Configuration and Reference, Volume 3* publication.

Example

In the following example, split horizon is disabled on a serial link connected to an X.25 network.

```
interface serial 0
encapsulation x25
no clns split-horizon
```

Setting Timing Parameters

The basic timing parameters for ISO-IGRP are adjustable. Because the ISO-IGRP routing protocol executes a distributed, asynchronous routing algorithm, it is important that these timers be the same for all routers in the network.

Use the following router subcommand to configure ISO-IGRP timers.

timers basic *update holddown flush*
no timers basic *update holddown flush*

The argument *update-interval* is the time, in seconds, between the sending of routing updates. The default value is 90 seconds.

The argument *holddown-interval* is the amount of time, in seconds, a system or area router is kept in holddown state, during which routing information regarding better paths is suppressed. (A router enters into a holddown state when an update packet is received that indicates the route is unreachable. The route is marked inaccessible and advertised as unreachable. However, the route is still used for forwarding packets.) When the holddown interval expires, routes advertised by other sources are accepted and the route is no longer inaccessible. The default value is 145 seconds.

The argument *flush* is the amount of time, in seconds, that a route remains in the routing table after it has been determined that it is not reachable. After that length of time, the route is removed from the routing table. The default value is 135 seconds.

Example

In the following example, updates are broadcast every 60 seconds. When an update packet is received that indicates the router is unreachable, the router will be in holddown state for 100 seconds before once more becoming accessible. If a router is not heard from in 130 seconds, the route is removed from the routing table.

```
router iso-igrp
timers basic 60 100 130
```

ISO-IGRP Metric Adjustments

The METRIC WEIGHTS command allows you to use different metrics for the IGRP routing protocol on IP and for ISO-IGRP on CLNS. Use the following router subcommand to configure the metric constants used in the ISO-IGRP composite metric calculation of reliability and load:

metric weights *qos k1 k2 k3 k4 k5*
no metric weights

The argument *qos* indicates QOS, or Quality of Service. QOS defines transmission quality and availability of service. The argument *qos* must be 0, the *default metric*. The *k1*, *k2*, *k3*, *k4*, and *k5* values apply to ISO-IGRP for the default metric QOS. The *k* values are metric constants used in the ISO-IGRP equation that converts an IGRP metric vector into a scalar quantity. The *k* values are numbers from 0 to 127; higher numbers mean a greater multiplier affect. The default *k* values are 1, 0, 1, 0, and 0 respectively.

Use the NO METRIC WEIGHTS command to return the five *k* constants to their default values.

For more information about adjusting IGRP metrics, see the *DECbrouter 90 Products Configuration and Reference, Volume 2* publication.

Example

In the following example, all five metric constants are set.

```
router iso-igrp
metric weights 0 2 0 1 0 0
```

Configuring Other ISO-IGRP Metrics

Two additional ISO-IGRP metrics can be configured. These are the bandwidth and delay associated with an interface. Refer to Chapter 8 of the *DECbrouter 90 Products Configuration and Reference Volume 1* publication for details about the BANDWIDTH and DELAY interface subcommands used to set these metrics.

Note

Using the BANDWIDTH and DELAY interface subcommands to change the values of the ISO-IGRP metrics will also change the values of IP IGRP metrics.

Configuring IS-IS CLNS Dynamic Routing

This section describes the global and interface commands used to configure ISO CLNS IS- IS routing support on the router. Each command description includes a simple example of the command. Examples set in the context of application configurations are provided in the subsequent section CLNS, ISO-IGRP, and CLNS IS-IS Routing Examples.

Note

ISO-IGRP and IS-IS should not be configured for the same area. This means that the clns net subcommand should not specify an NSAP address where all bytes up to (but not including) the System ID are the same when enabling both ISO-IGRP and IS-IS routing. See Chapter 1 of the *DECbrouter 90T Configuration and Reference Volume 3* for a description of the CLNS NET subcommand.

Enabling IS-IS Routing

Use the following global configuration command to enable the IS-IS routing protocol:

router isis [*tag*]
no router isis [*tag*]

The argument *tag* defines a meaningful name for a routing process. For example, you could define a routing process named *Finance*. The argument *tag* is optional; if it is not specified, a null tag is assumed. You can specify only one IS-IS process per router.

Note

The *tag* argument must be unique among all CLNS router processes for a given router.

The NO ROUTER ISIS global configuration command with the appropriate tag disables IS-IS routing for the system.

Example

The following example illustrates starting IS-IS routing with the optional tag argument:

```
router isis Finance
```

Configuring Network Entity Titles

Network Entity Titles (NETs) define the area addresses for the IS-IS area, as described in Chapter 1 of the *DECbrouter 90 Products Configuration and Reference, Volume 3* publication.

Use the following router subcommand to configure a Network Entity Title (NET) for the routing process:

net *network-entity-title*
no net *network-entity-title*

The argument *network-entity-title* is the NET that specifies the area address and the system ID for an IS-IS routing process. This argument can be either an address or a name. For IS- IS, multiple NETs per router are allowed, with a maximum of three. There is no default value for this command.

Note

Although IS-IS allows you to configure multiple NETs, ISO-IGRP allows only one NET per router.

The NO NET command removes a specific NET.

Examples

The following example illustrates specifying a single NET:

```
router isis Finance
net 47.0004.004d.0001.0000.0c11.1111.00
```

The following example illustrates the use of a name for a NET:

```
clns host NAME 39.0001.0000.0c00.1111.00
!
router isis
net NAME
!
```

Specifying Router Level Support

Use the following router subcommand to configure the IS-IS level at which the router will operate:

is-type [**level-1** | **level-1-2** | **level-2-only**]
no is-type [**level-1** | **level-1-2** | **level-2-only**]

If **level-1-2** is specified, the router acts as a station router. If **level-1-2** is specified, the router acts as both a station router and an area router. If **level-2-only** is specified, the router acts as an area router only. The default value is **level-1-2**.

The command NO IS-TYPE resets the parameter to the default.

Example

The following example specifies an area router:

```
is-type level-2-only
```

Redistributing Static Routes

Use the following router subcommand to redistribute static routes into IS-IS:

redistribute static [**clns**]
no redistribute static [**clns**]

The default is **redistribute static clns**.

Importing Routes Learned by Other CLNS Routing Protocols

If you have a router that is in two routing domains, you may want to redistribute routing information between the two domains.

Use the following router configuration subcommand to specify the dynamic routes to be redistributed into the IS-IS domain:

redistribute *CLNS-routing-protocol* [*tag*]
no redistribute *CLNS-routing-protocol* [*tag*]

The argument *CLNS-routing-protocol* specifies the routing protocol that is redistributed into IS-IS. When redistributing CLNS prefix routes, only an CLNS routing protocol name or the keyword **static** is allowed. Supported *CLNS-routing-protocol* keywords are **static** and **iso-igrp**. Static routes are always redistributed into IS-IS unless a NO REDISTRIBUTE STATIC command is executed. Redistribution only occurs for level-2 routing.

The optional argument *tag* is the name of a process.

The REDISTRIBUTE global configuration command causes the routes learned by routing process *tag* to be advertised into IS-IS.

Example

The following example illustrates redistribution of ISO-IGRP routes of Michigan and ISO-IGRP routes of Ohio into the IS-IS area tagged Illinois.

```
router isis Illinois
redistribute iso-igrp Michigan
redistribute iso-igrp Ohio
```

Configuring IS-IS for an Interface

Use the following interface subcommand to enable IS-IS routing for CLNS on a specific interface:

clns router isis *tag*
no clns router isis *tag*

Use the same process name for the argument *tag* as specified in the global configuration command ROUTER ISIS described earlier in this chapter.

Example

In the following example, IS-IS routing is enabled on interface serial 0.

```
router isis finance
interface serial 0
clns router isis finance
```

Exporting IS-IS Routes into Other Protocols

Use the following router subcommand for importing IS-IS routes into other protocols:

> **redistribute isis** [*tag*]
> **no redistribute isis** [*tag*]

The optional argument *tag* defines a meaningful name for a routing process. The default setting is NO REDISTRIBUTE ISIS.

Example

In the following example, the process named manufacturing is configured to be imported into other protocols.

```
router iso-igrp finance
redistribute isis manufacturing
```

Specifying Preferred Routes

Use the following router subcommand to configure the administrative distance for CLNS routes learned:

> **distance** *value* [**clns**]
> **no distance** *value* [**clns**]

The argument *value* is the administrative distance, indicating the trustworthiness of a routing information source. This argument has a numerical value between 0 and 255. A higher relative value indicates a lower trustworthiness rating. Preference is given to routes with smaller values.

The default values are as follows:

- Static routes—10
- ISO-IGRP routes—100
- IS-IS routes—110

The keyword **clns** is optional. The default keyword is **clns**.

Example

In the following example, the distance value for CLNS routes learned is 90. Preference is given to these CLNS routes versus routes with the default administrative distance value of 110.

```
router isis
distance 90 clns
```

Configuring IS-IS Link State Metrics

Use the following interface subcommand to configure the metric (or cost) for a specified interface:

isis metric *default-metric* [**level-1** | **level-2**]
no isis metric *default-metric* [**level-1** | **level-2**]

The optional keyword/argument pair **metric** *default-metric* specifies the link state cost assigned to the interface. The *default-metric* argument is a dimensionless link state cost, formed as a 24-bit decimal number. The default value for the *default-metric* argument is 10. You can configure this metric for Level 1 and/or Level 2 routing.

The NO ISIS METRIC command resets the *default-metric* to ten. Specification of the **level-1** or **level-2** optional keywords resets the metric only for Level 1 or Level 2 routing, respectively.

Example

In the following example, interface serial 0 is configured for a default link-state metric cost of 15 for level 1.

```
interface serial 0
isis metric 15 level-1
```

Setting the Advertised hello Interval

Use the following interface subcommand to specify the length of time, in seconds, between hello packets the router sends on the interface:

isis hello-interval *seconds* [**level-1** | **level-2**]
no isis hello-interval *seconds* [**level-1** | **level-2**]

The argument *seconds* is an unsigned integer value. A value three times the hello interval *seconds* is advertised as the *holdtime* in the hello packets transmitted. With smaller hello intervals, topological changes are detected faster, but there is more routing traffic.

The hello interval can be configured independently for level-1 and level-2, except on serial point-to-point interfaces. (Because there is only a single type of hello packet sent on serial links, it is independent of level-1 or level-2.) The **level-1** and **level-2** keywords are used on X.25, SMDS, and Frame Relay multiaccess networks. The default value is 10 seconds.

Example

In the following example, interface serial 0 is configured to advertise hello packets every five seconds. The router is configured to act as a station router. This will cause more traffic than configuring a longer interval, but topological changes will be detected faster.

```
interface serial 0
isis hello-interval 5 level-1
```

Setting the Advertised CSNP Interval

Complete Sequence Number PDUs (CSNP)s are sent by the designated router to maintain database synchronization. Use the following interface command to configure the IS-IS CSNP interval for the interface:

isis csnp-interval *seconds* [**level-1** | **level-2**]
no isis csnp-interval *seconds* [**level-1** | **level-2**]

This *seconds* argument is the interval of time between transmission of CSNPs on multiaccess networks. This interval only applies to the designated router and can be configured independently for level-1 and level-2. This command does not apply to serial point-to-point interfaces. This command does apply to WAN connections if the WAN is viewed as a multiaccess meshed network. The default value for *seconds* is 10 seconds. The **level-1** and **level-2** keywords are used on X.25, SMDS, and Frame Relay multiaccess networks.

Example

In the following example, interface serial 0 is configured to transmit CSN PDUs every five seconds. The router is configured to act as a station router.

```
clns router isis
interface serial 0
isis csnp-interval 5 level-1
```

Setting the Retransmission Interval

The command described in this section is used to configure the amount of time the router will wait before retransmitting packets to an unresponsive router.

Use the following interface command to configure the number of seconds between retransmission of IS-IS link state PDU (LSP) retransmission for point-to-point links.

isis retransmit-interval *seconds*
no isis retransmit-interval *seconds*

The value for the *seconds* argument is an integer that should be greater than the expected round-trip delay between any two routers on the attached network. The setting of this parameter should be conservative, or needless retransmission will result. The value should be larger for serial lines and virtual links. The default value is five seconds.

Example

The following example configures interface serial 0 for retransmission of IS-IS LSP every 10 seconds for a large serial line.

```
clns router isis
serial interface 0
isis retransmit-interval 10
```

Specifying Designated Router Election

Use the following interface subcommand to configure the priority of designated routers.

isis priority *value* [**level-1** | **level-2**]
no isis priority [**level-1** | **level-2**]

The acceptable range of values is 0 to 127. The default *value* is 64. Priorities can be configured for Level 1 and Level 2 individually. The NO ISIS PRIORITY command resets priority to 64. Specification of the **level-1** or **level-2** optional keywords resets priority only for Level 1 or Level 2 routing, respectively.

Example

The following example shows level 1 routing given priority by setting the priority level to 50 (default is 64).

```
clns router isis
interface serial 0
isis priority 50 level-1
```

Specifying Interface Circuit Type

Use the following interface subcommand to configure the type of *adjacency* desired for this interface:

isis circuit-type [**level-1**] [**level-1-2**] [**level-2-only**]
no isis circuit-type

If **level-1** is specified, a Level 1 adjacency is established if there is at least one area address in common between this system and its neighbors.

If **level-1-2** is specified, a Level 1 and 2 adjacency is established if the neighbor is also configured as **level-1-2** and there is at least one area in common. If there is no area in common, a Level 2 adjacency is established.

If **level-2-only** is specified, a Level 2 adjacency is established if and only if the neighbor is configured exclusively to be a Level 2 router.

The default value for this command is LEVEL-1-2. The NO ISIS CIRCUIT-TYPE command resets the circuit type to Level l and Level 2.

Example

In the following example, a router is configured to require level 1 adjacency if there is at least one area address in common between this system and its neighbors.

```
clns router isis
interface serial 0
isis circuit type level-1
```

Configuring IS-IS Authentication Passwords

Using the authentication password commands in this section, you can assign passwords to interfaces, areas, and domains.

Assigning a Password for an Interface

Use the following interface subcommand to configure the authentication password for an interface:

isis password *password* [**level-1** | **level-2**]
no isis password [**level-1** | **level-2**]

Different passwords can be assigned for different routing levels using the optional **level-1** and **level-2** keyword arguments. By default authentication is disabled.

The noisispassword command disables authentication for IS-IS. Specifying **level-1** or **level-2** optional keywords disables the password only for Level 1 or Level 2 routing, respectively. If no keyword is specified, the default is **level-1**.

Example

The following example configures a password for interface serial 0 at level 1.

```
clns router isis
interface serial 0
isis password frank level-1
```

Assigning a Password for an Area

Use the following router subcommand to configure the area authentication password:

area-password [*password*]
no area-password [*password*]

The default is **no area-password** *password*. This password is inserted in level-1 (station router level) LSPs.

Example

The following example assigns an area authentication password.

```
clns router isis
area-password angel
```

Assigning a Password for a Domain

Use the following router subcommand to configure the routing domain authentication password:

domain-password [*password*]
no domain-password [*password*]

The default is **no domain-password** *password* no domain-password password. This password is inserted in level-2 (the area router level) LSP.

Example

The following example assigns an authentication password to the routing domain.

```
clns router isis
domain-password flower
```

CLNS, ISO-IGRP, and CLNS IS-IS Routing Examples

This section provides configuration examples of both intra- and interdomain static and dynamic routing using static, ISO-IGRP, and IS-IS routing techniques.

Basic Static Routing

Configuring Ethernets and serial lines for CLNS can be as simple as just enabling CLNS on the interfaces. This is all that is ever required on serial lines using HDLC encapsulation. If all systems on an Ethernet support ISO 9542 ES-IS, nothing else is required and an Ethernet and a serial line can be configured as in the following example:

```
clns net 47.0004.004D.0055.0000.0C00.BF3B.00
clns routing
interface ethernet 0
clns enable
interface serial 0
clns enable
clns route 47.0004.004d.0099 serial 0
clns route 47.0005 serial 0
```

Static Routing

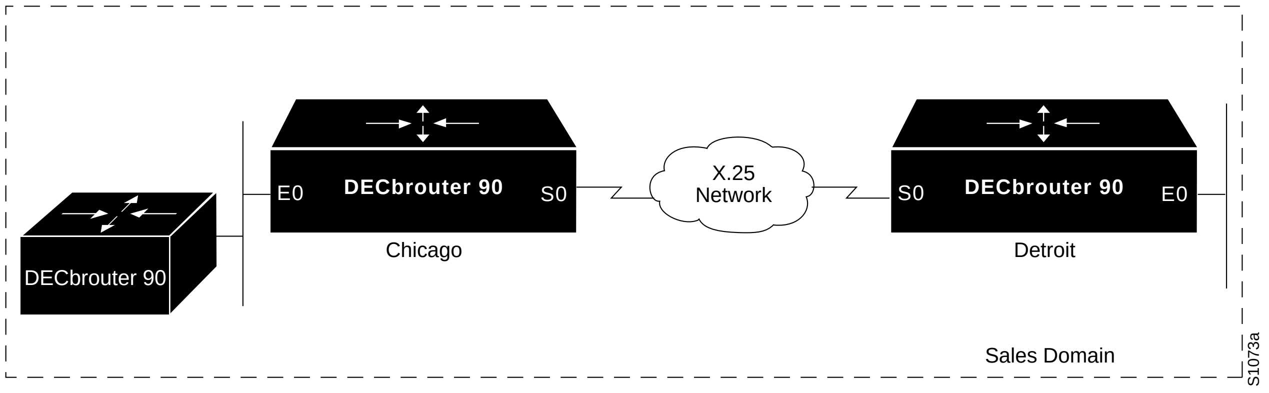
The following is a more complete example of CLNS static routing on a system with two Ethernet interfaces. After configuring routing, define an NET and enable CLNS on the Ethernet 0 and Serial 0 interfaces. You must then define an IS-neighbor and define a static route with the CLNS ROUTE command, as shown. In this situation, there is an IS on Ethernet 0 that does not support ES-IS.

```
clns host george 39.0001.0000.0c00.1111.00
clns net george
clns routing
interface Serial 0
clns enable
interface Ethernet 0
clns enable
clns is-neighbor george 0000.0C00.62e7
clns route 47.0004.000c foo
```

Static Intradomain Routing

Figure 2–1 and the configurations that follow demonstrate how to use static routing inside a domain. Imagine a company with two branch offices in Detroit and Chicago, connected with an X.25 link. These offices are both in the domain named Sales.

Figure 2–1 CLNS X.25 Intradomain Routing



The following is one way to configure the router in Chicago:

```
clns host chicago 47.0004.0050.0002.0000.0c00.243b.00
clns host detroit 47.0004.0050.0001.0000.0c00.1e12.00
clns routing
router iso-igrp sales
net chicago
interface ethernet 0
clns router iso-igrp sales
interface serial 0
encapsulation x25
x25 address 31342174523156
x25 nvc 4
clns router iso-igrp sales
clns is-neighbor detroit 31343136931281 broadcast
```

This configuration will bring up an X.25 virtual circuit between the router in Chicago and the router in Detroit. Routing updates will be sent across this link. This implies that the virtual circuit could be up continuously.

If you do not want the virtual circuit to be up continuously, use the following configuration instead.

```
!
clns host chicago 47.0004.0050.0002.0000.0c00.243b.00
clns host detroit 47.0004.0050.0001.0000.0c00.1e12.00
router iso-igrp sales
net 47.0004.0060.0002.0000.0c00.243b.00
!
interface ethernet 0
clns router iso-igrp sales
!
interface serial 0
encapsulation x25
x25 address 31342174523156
x25 nvc 4
clns enable
clns is-neighbor detroit 31343136931281
!
clns route 47.0004.0050.0001 detroit
!
```

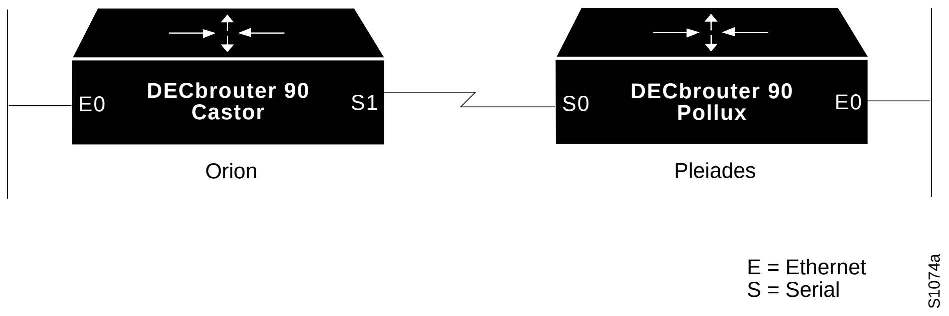
If the Chicago office should grow to contain multiple routers, it would be appropriate for each of those routers to know how to get to Detroit. Add the following command to redistribute information between routers in Chicago:

```
router iso-igrp sales
redistribute static
```

Static Interdomain Routing

Figure 2–2 and the example configurations that follow illustrate how to configure two routers that distribute information across domains. In this example, Castor and Pollux communicate across a serial link.

Figure 2–2 CLNS Interdomain Static Routing



Example Configuration for Castor

```
router iso-igrp orion
net 47.0006.0200.0100.0102.0304.0506.00
!
clns host pollux 47.0006.0200.0200.1112.1314.1516.00
!
interface ethernet 0
clns router iso-igrp orion
!
interface serial 1
clns enable
!
clns route 39.0001 pollux
```

Example Configuration for Pollux

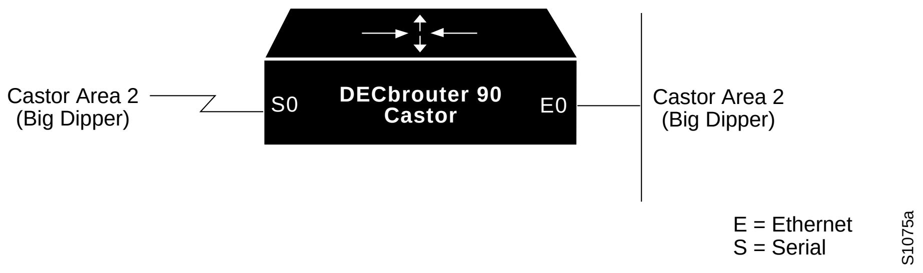
```
router iso-igrp pleiades
net 47.0006.0200.0200.1112.1314.1516.00
!
clns host castor 47.0006.0200.0100.0001.0102.0304.0506.00
!
interface ethernet 0
clns router iso-igrp pleiades
!
interface serial 0
clns enable
!
clns route 47.0006.0200.0100 castor
```

CLNS routing updates will not be sent on the serial link; however, CLNS packets will be sent and received over the serial link.

Routing Within the Same Area

Figure 2–3 and the example configuration that follows illustrate how to configure dynamic routing within a routing domain. The router can exist in one or more areas within the domain. The router named Castor exists in a single area.

Figure 2–3 CLNS Dynamic Routing Within a Single Area

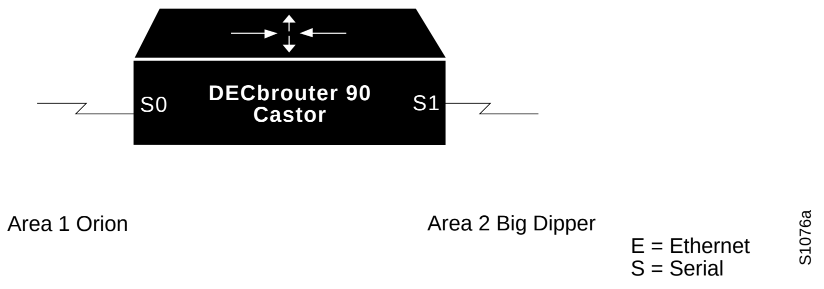


```
clns routing
router iso-igrp bigdipper
net 47.0004.004D.0002.0000.0C00.0506.00
interface Ethernet 0
clns router iso-igrp bigdipper
interface Serial 0
clns router iso-igrp bigdipper
```

Dynamic Routing in More Than One Area

Figure 2–4 and the example configuration that follows illustrate how to configure a router named Castor that exists in two areas.

Figure 2–4 CLNS Dynamic Routing Within Two Areas

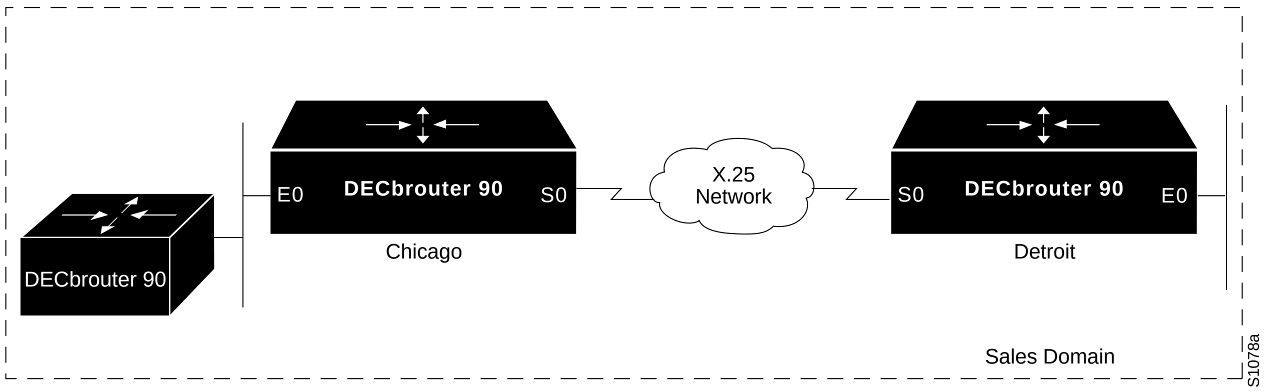


```
clns routing
router iso-igrp orion
net 47.0004.004D.0001.212223242526.00
router iso-igrp bigdipper
net 47.0004.004D.0002.212223242526.00
interface serial 0
clns router iso-igrp orion
interface serial 1
clns router iso-igrp bigdipper
```

Dynamic Interdomain Routing

Figure 2–5 and the examples that follow illustrate how to configure three domains to be transparently connected.

Figure 2–5 CLNS Interdomain Dynamic Routing



Example Configuration for Castor Configuration for Router X

```
clns routing
router iso-igrp A
net 47.0006.0200.0002.0102.0104.0506.00
redistribute iso-igrp B
router iso-igrp B
net 47.0007.0200.0003.0102.0104.0506.00
redistribute iso-igrp A
interface serial 0
clns router iso-igrp A
interface serial 1
clns router iso-igrp B
```

Example Configuration for Castor Configuration for Router Y

```
clns routing
router iso-igrp B
net 47.0007.0200.0004.0102.0104.0506.00
redistribute iso-igrp C
router iso-igrp C
net 47.0008.0200.0005.0102.01040.506.00
redistribute iso-igrp B
interface serial 0
clns router iso-igrp B
interface serial 1
clns router iso-igrp C
```

Router X will inject a prefix route for domain A into domain B. Domain B will inject this prefix route and a prefix route for domain B into domain C.

You also can configure a border router between domain A and domain C.

CLNS IS-IS Routing Configuration

The examples that follow illustrate the basic syntax and configuration command sequence for IS-IS routing.

Example Configuration for Castor 1: Level 1 and Level 2 Routing

The following example illustrates using the IS-IS protocol to configure a single area address for Level 1 and Level 2 routing.

```
clns routing
router isis
net 47.0004.004d.0001.0000.0c00.1111.00
interface Ethernet 0
clns router isis
interface Serial 0
clns router isis
interface Serial 1
clns router isis
```

Example Configuration for Castor 2: Level 2 Routing Only

The following example illustrates a similar configuration featuring a single area address being used for specification of Level 1 and Level 2 routing; however, in this case, interface Serial 0 is configured for Level 2 routing only.

```
clns routing
router isis
net 47.0004.004d.0001.0000.0c00.1111.00
interface Ethernet 0
clns router isis
interface Serial 0
clns router isis
isis circuit-type level-2-only
interface Serial 1
clns router isis
```

Example 3: CLNS Configuration

The following example illustrates an CLNS configuration example. In this example, IS-IS runs with two area addresses, metrics tailored, and different circuit types specified for each interface.

```
clns routing
router isis area1
net 47.0004.004d.0001.0000.0c11.1111.00
net 47.0004.004d.0011.0000.0c11.1111.00
is-type level-1-2
interface Ethernet 0
clns router isis area1
isis metric 5 level-1
isis circuit-type level-1
interface Serial 1
clns router isis area1
isis metric 2 level-2
isis circuit-type level-2-only
interface serial 0
clns router isis area1
isis circuit-type level-1-2
isis priority 3 level-1
isis priority 1 level-2
```

Example 4: ISO CLNS Dynamic Route Redistribution

The following example illustrates route redistribution between IS-IS and ISO-IGRP domains. In this case, the IS-IS domain is on interface Ethernet 0; the ISO-IGRP domain is on interface Serial 0. The IS-IS routing process is assigned a null tag; the ISO-IGRP routing process is assigned a tag of *remote-domain*.

```
router isis
net 39.0001.0001.0000.0c00.1111.00
redistribute iso-igrp remote-domain

router iso-igrp remote-domain
net 39.0002.0001.0000.0c00.1111.00
redistribute isis

interface Ethernet 0
clns router isis

interface Serial 0
clns router iso-igrp remote
```

Monitoring an IS-IS CLNS Network

Use the EXEC commands described in this section to obtain displays of activity on the ISO CLNS network.

Displaying Destination Routing Table

Use the WHICH-ROUTE command if you want to know which next-hop router will be used or if you have multiple processes running and want to troubleshoot your configuration.

Use the following EXEC command to display the routing table in which the specified CLNS destination is found:

which-route *{NSAP-address | CLNS-name}*

The argument *NSAP-address* is the specified CLNS destination network address. The argument *CLNS-name* is the destination host name. See Chapter 1 of the *DECbrouter 90T Configuration and Reference Volume 3* publication for a complete description of the CLNS HOST command.

Route information can reside in the following tables:

- IS-IS level-1 routing table
- ISO-IGRP system-id or area routing table
- Prefix routing table (IS-IS level-2 routes, ISO-IGRP domain routes, and static routes)
- Adjacency database

Examples

The following example display shows that destination information for router "gray" is found in the IS-IS level-1 routing table. The destination is on the local system.

```
gray# which-route gray
Route look-up for destination 39.0001.0000.0c00.bda8.00, GRAY
  Found route in IS-IS level-1 routing table - destination is local
```

The following example display show that destination information for NSAP address 49.0001.0000.0c00.bda8.00 is found in the ISO-IGRP level-1 routing table. The destination is on the local system.

```
gray# which-route 49.0001.0000.0c00.bda8.00
Route look-up for destination 49.0001.0000.0c00.bda8.00
  Found route in ISO-IGRP routing table - destination is local
```

The following example display shows that destination information for router "green" is found in the IS-IS level-1 routing table. The destination is not on the local system and Table 2–3 describes the display fields in the adjacency entry used to reach system "green."

```
gray# which-route green
Route look-up for destination 39.0001.0000.0c00.7f06.00, GREEN
  Found route in IS-IS level-1 routing table

Adjacency entry used:
System Id       SNPA             Interface   State  Holdtime  Type  Protocol
GREEN           0000.0c00.2d55   Ethernet0   Up     91        L1L2  IS-IS
  Area Address(es): 39.0001
```

Table 2–3 Which-Route Field Descriptions

System ID	The six-byte value that identifies a system in an area. A name is displayed in this field if one has been assigned with the clns host command.
SNPA	SNPA data link address
Interface	Interface from which system information was learned.
State	State of the ES or IS. Possible values are as follows: Init—The system is an IS and is waiting for an IS- IS Hello message. The neighbor to the IS-IS is not adjacent. Up—The ES or IS is reachable.
Holdtime	The number of seconds for which the information is valid.
Type	Adjacency type. Possible values are as follows: ES—An end-system adjacency that is either discovered by the ES-IS protocol or statically configured. IS—A router adjacency that is either discovered by the IS-IS protocol or is statically configured. L1—A router adjacency for Level-1 routing only. L1L2—A router adjacency for Level-1 and Level-2 routing. L2—A router adjacency for Level-2 only.
Protocol	Protocol through which the adjacency was learned. Valid protocol sources are ES-IS, IS-IS, ISO-IGRP, and Static.

The following example display shows that destination information for NSAP address 49.0001.1111.1111.1111.00 is found in the ISO-IGRP routing table. Table 1-3 describes the display fields in the adjacency entry used to reach NSAP address 49.0001.1111.1111.1111.00.

```
gray# which-route 49.0001.1111.1111.1111.00
Route look-up for destination 49.0001.1111.1111.1111.00
  Found route in ISO-IGRP routing table

Adjacency entry used:
System Id       SNPA             Interface   State  Holdtime  Type  Protocol
1111.1111.1111  0000.0c01.151d   Ethernet0 Up     38        L1L2  ISO-IGRP
  Area Address(es): 49.0001
```

The following example display indicates that the specified address is not found in a routing table.

```
gray# which-route 47.0003.0000.0000.0000.00
Route look-up for destination 47.0003.0000.0000.0000.00
  Route not found
```

The following example display indicates that the specified NSAP address was found in the CLNS prefix routing table. This information is followed by the route entry used to reach NSAP address 49.0003.0000.0000.0000.00.

```
gray# which-route 49.0003.0000.0000.0000.00
Route look-up for destination 49.0003.0000.0000.0000.00
  Found route in CLNS prefix routing table

Route entry used:
49 [10/0]
  via 1111.1111.1111, Ethernet0, Static
```

Displaying the IS-IS Level 1 Routing Table

The SHOW ISIS ROUTE EXEC command displays the IS-IS Level 1 routing table. The command syntax is as follows:

show isis route

Following is sample output:

```
Chrysostom> show isis route
IS-IS Level-1 Routing Table - Version 34
System Id      Next-Hop SNPA Interface Metric State
0000.0C00.0C35 0000.0C00.0C35 0000.0c00.0c36 Ethernet0 20 Up
0800.2B16.24EA 0800.2B16.24EA aa00.0400.2d05 Ethernet0 10 Up
0800.2B14.060E 0800.2B14.060E aa00.0400.9205 Ethernet0 10 Up
0800.2B14.0528 0800.2B14.0528 aa00.0400.9105 Ethernet0 10 Up
0000.0C00.40AF 0000.0000.0000 -- -- 0 Up
0000.0C00.62E6 0000.0C00.62E6 0000.0c00.62e7 Ethernet0 10 Up
AA00.0400.2D05 0800.2B16.24EA aa00.0400.2d05 Ethernet0 10 Up
```

This display presents the Level 1 forwarding table for IS-IS learned routes. The following fields are provided in this display:

- System ID—The identification value of the system listed in Level 1 forwarding table. A name is displayed in this field if one has been assigned with the CLNS HOST command.
- Next-Hop—System ID of best cost next-hop to listed address. A name is displayed in this field if one has been assigned with the CLNS HOST command.
- SNPA—SNPA of next-hop system
- Interface—Interface through which next-hop system is known by router
- Metric—IS-IS metric for the route
- State—Up (active) or Down (nonoperational)

Displaying Protocol-Specific Information

The SHOW CLNS PROTOCOL command lists the protocol-specific information for each ISO-IGRP routing process in this router. Enter this command at the EXEC prompt:

show clns protocol [*domain* | *area-tag*]

There always will be at least two routing processes, a Level 1 and a Level 2, and there may be more. The optional argument *domain* specifies a particular ISO-IGRP routing domain; the optional argument *area-tag* specifies a particular IS-IS area.

Following is sample output:

```
Chrysostom> show clns protocol
ISO-IGRP Level 1 Router: remote
  Routing for domain: 39.0003 area: 0020
  Sending Updates every 45 seconds. Next due in 11 seconds
  Invalid after 135 seconds,
  Hold down for 145 seconds
  Sending Router Hellos every 17 seconds. Next due in 9 seconds
  Invalid after 51 seconds,
  IGRP metric weight K1=1, K2=0, K3=1, K4=0, K5=0
  Interfaces in domain/area:
        Ethernet0
--More--
ISO-IGRP Level 2 Router: DOMAIN_remote
  Routing for domain: 39.0003
  Redistribute:
    isis (Null Tag)
  Sending Updates every 45 seconds. Next due in 2 seconds
  Invalid after 135 seconds,
  Hold down for 145 seconds
  Sending Router Hellos every 17 seconds. Next due in 0 seconds
  Invalid after 51 seconds,
  ISO-IGRP metric weight K1=1, K2=0, K3=1, K4=0, K5=0
  Interfaces in domain/area:
        Ethernet0
 --More--
IS-IS Router: <Null Tag>
  System Id: 0000.0C00.224D.00  IS-Type: level-1-2
  Manual area address(es):
        39.0004.0030
  Routing for area address(es):
        39.0004.0030
  Interfaces supported by IS-IS:
        Serial1
  Next global update in 530 seconds
  Redistributing:
    static
    iso-igrp (remote)

Chrysostom> show clns protocol

IS-IS Router: <Null Tag>
System Id: 1600.8906.4023.00  IS-Type: level-1-2
Manual area address(es):
 47.0005.80ff.ef00.0000.0001.5940
Routing for area address(es):
 47.0005.80ff.ef00.0000.0001.5940
Interfaces supported by IS-IS:
     Ethernet0 - OSI
 Tunnel2 - OSI
 Tunnel0 - OSI
Next global update in 37 seconds
Redistributing:
 static
Distance: 110
```

In the ISO-IGRP portion of the display:

- The first line provides the domain address and area number for Level 1 routing processes. For Level 2 routing processes, this command lists the domain address.
- The next set of fields indicate some of the protocol timers. The field labeled "Sending updates" displays when the next routing updates will be sent.
- The Invalid field indicates how long routing updates are to be believed.
- The Hold Down field indicates how long a route will be held down before new information is to be believed.
- The Sending Router Hellos field indicates how often the routers will send Hello packets to each other and when the next is due.
- The field labeled Invalid indicates how long a neighbor entry will be remembered.
- The ISO-IGRP metric weight display lists the weights applied to the various components of the metric. These fields are followed by the list of interfaces that are in this area.

In the IS-IS portion of the display:

- The first portion specifies the relevant IS-IS area tag.
- The second line lists the System ID and IS Type.
- The next two information fields display area addresses.
- Area addresses are followed by a list of interfaces on the router supporting IS-IS.
- Several information fields are provided that indicate the next expected IS-IS update and any configuration of route redistribution.

Displaying the IS-IS Database

The SHOW ISIS DATABASE EXEC command displays the IS-IS link state database. If **detail** is specified, the contents of each LSP is displayed. Otherwise, a summary display is provided. The command syntax is as follows:

show isis database [**level-1** | **level-2** | **l1** | **l2** | **detail**]

Note

The notations **l1** and **l2** are abbreviations for the options **level-1** and **level-2** , respectively. Each of the options shown in brackets for this command can be entered in a arbitrary string within the same command entry. For example, the following are both valid command specifications and provide the same display: SHOW ISIS DATA DETAIL L2 and SHOW ISIS DATA L2 DETAIL.

Following is sample output of the command with no options or as SHOW ISIS DATA L1 L2:

```
IS-IS Level-1 Link State Database
LSPID                 LSP Seq Num  LSP Checksum  LSP Holdtime  ATT/P/OL
0000.0C00.0C35.00-00  0x0000000C   0x5696        792           0/0/0
0000.0C00.40AF.00-00* 0x00000009   0x8452        1077          1/0/0
0000.0C00.62E6.00-00  0x0000000A   0x38E7        383           0/0/0
0000.0C00.62E6.03-00  0x00000006   0x82BC        384           0/0/0
0800.2B16.24EA.00-00  0x00001D9F   0x8864        1188          1/0/0
0800.2B16.24EA.01-00  0x00001E36   0x0935        1198          1/0/0

IS-IS Level-2 Link State Database
LSPID                 LSP Seq Num  LSP Checksum  LSP Holdtime  ATT/P/OL
0000.0C00.0C35.03-00  0x00000005   0x04C8        792           0/0/0
0000.0C00.3E51.00-00  0x00000007   0xAF96        758           0/0/0
0000.0C00.40AF.00-00* 0x0000000A   0x3AA9        1077          0/0/0
```

This display presents IS-IS related information for the link state database. The following fields are provided in this display:

- LSPID—The link state PDU ID. The first six octets form the system ID. A name is displayed in this field if one has been assigned with the CLNS HOST command. The next octet is the pseudo ID. When this value is zero, the LSP describes links from the system. When it is nonzero, the LSP is a pseudonode LSP. The designated router for an interface is the only system that originates pseudonode LSPs. The last octet is the LSP number. If there is more data than can fit in a single LSP, additional LSPs are sent with increasing LSP numbers. An asterisk (*) indicates that the LSP was originated by the local system.
- LSP Seq Num—The sequence number for the LSP that allows other systems can determine they have received the latest information from the source.
- LSP Checksum—The checksum of the entire LSP packet.
- LSP Holdtime—The amount of time the LSP remains valid, in seconds.
- ATT— The attach bit. This indicates that the router is also a Level 2 router, and it can reach other areas.
- P—The P bit. Detects whether the IS is area partition repair capable.
- OL—The Overload bit. Determines whether the IS is congested.

A sample output of the SHOW ISIS DATABASE DETAIL command follows.

```
IS-IS Level-1 Link State Database
LSPID                 LSP Seq Num  LSP Checksum  LSP Holdtime  ATT/P/OL
SNUG.00-00          * 0x00000006   0xF6FA        288           0/0/0
  Area Address: 47.0004.004D.0001
  Metric: 10   IS DEMETER.00
  Metric: 0    ES SNUG
 --More--
DEMETER.00-00         0x0000000D   0x9106        330           0/0/0
  Area Address: 47.0004.004D.0001
  Metric: 10   IS DEC.01
  Metric: 10   IS SNUG.00
  Metric: 0    ES DEMETER
 --More--
DEC.00-00             0x00000F98   0xC041        720           1/0/0
  Area Address: 47.0004.004D.0001
  Metric: 10   IS DEC.01
  Metric: 0    ES DEC
  Metric: 0    ES AA00.0400.2D05
 --More--
DEC.01-00             0x00000FB3   0x14BB        334           1/0/0
  Metric: 0    IS DEC.00
  Metric: 0    IS DEMETER.00
  Metric: 0    ES AA00.0400.9105
  Metric: 0    ES 0800.2B14.0528
  Metric: 0    ES AA00.0400.9205
  Metric: 0    ES 0800.2B14.060E

IS-IS Level-2 Link State Database
LSPID                 LSP Seq Num  LSP Checksum  LSP Holdtime  ATT/P/OL
0000.0C00.0C35.03-00  0x00000005   0x04C8        317           0/0/0
  Metric: 0    IS 0000.0C00.0C35.00
 --More--
0000.0C00.3E51.00-00  0x00000009   0xAB98        1182          0/0/0
  Area Address: 39.0004
  Metric: 10   IS 0000.0C00.40AF.00
  Metric: 10   IS 0000.0C00.3E51.05
 --More--
0000.0C00.40AF.00-00* 0x0000000A   0x3AA9        599           0/0/0
  Area Address: 47.0004.004D.0001
  Area Address: 39.0001
  Metric: 10   IS 0800.2B16.24EA.01
  Metric: 10   IS 0000.0C00.3E51.00
```

In addition to the information displayed in SHOW ISIS DATABASE, this SHOW command displays the contents of each LSP.

Debugging IS-IS for CLNS Dynamic Routing Protocols

Use the EXEC commands described in this section to troubleshoot and monitor the ISO CLNS routing protocols. For each DEBUG command, there is a corresponding UNDEBUG command that turns the message logging off.

debug isis-adj-packets

Logs information for adjacency-related functions such as Hello packets sent and received and IS-IS adjacencies going up and down.

debug isis-spf-events

Logs events associated with the Dijkstra algorithm.

debug isis-update-packets
Logs incoming and outgoing sequence number packets and link state packets. It also logs flooding-related functions.

Global Configuration Command Summary

This section provides an alphabetical summary of the ISO CLNS global configuration commands.

[**no**] **clns route** *nsap-prefix* **discard**
Explicitly tells a router to discard packets with the specified *nsap-prefix* (full syntax listed).

[**no**] **clns route** *nsap-prefix interface-type unit* [*SNPA-address*]
Applies CLNS ROUTE command to a specific interface. The optional argument *SNPA-address* is required for multiaccess networks.

[**no**] **clns route** *nsap-prefix* {*next-hop-NET* | *name*}
Configures a static route for a router.

The argument *nsap-prefix* is the front portion of a Network Service Access Point (NSAP) address. The argument *next-hop-NET* is the Network Entity Title (NET) address of the node to which packets, or Protocol Data Units (PDUs), will be sent next. The argument *name* is the name of the node to which PDUs will be sent next. NSAPs that start with the value given for the argument *nsap-prefix* are forwarded to *next-hop-NET* or the next hop node *name*.

[**no**] **router isis** *tag*
Identifies the area the router will work in and lets the router know that it will be routing dynamically rather than statically. The keyword **isis** specifies dynamic routing using the IS-IS protocol. The argument *tag* defines a meaningful name for a routing process.

[**no**] **router iso-igrp** [*tag*]
Enables the ISO-IGRP routing process and identifies the address for the router. The keyword **iso-igrp** specifies dynamic routing using the ISO-IGRP protocol. You can specify up to ten ISO-IGRP processes. The argument *tag* defines a meaningful name for routing process.

Router Subcommand Summary

This section provides an alphabetical list of the ISO CLNS router subcommands.

[**no**] **area-password** [*password*]
Configures the authentication password for an area.

[**no**] **clns split-horizon**
Implements split horizon for ISO-IGRP updates. The default value is **split-horizon** for all LAN interfaces, and the default value for WAN interfaces on X.25, Frame Relay, or SMDS networks is **no split-horizon**.

[**no**] **distance** *value* [**clns**]
Assigns administrative distance for a particular routing protocol.
Default values:

- Static routes—10
- ISO-IGRP routes—100
- IS-IS routes—110

[**no**] **domain-password** [*password*]
Configures the routing domain authentication password.

[**no**] **is-type** [**level-1** | **level-1-2** | **level-2-only**]
Configures the level at which the router should operate. If **level-1** is specified, the router acts as an station router. If **level-1-2** is specified, the router acts as both an station router and a router that routes between areas. If **level-2-only** is specified, the router acts as an interarea router only. The default value is LEVEL-1-2. The NO IS-TYPE command resets routing level to Level 1 and 2.

[**no**] **metric weights** *qos k1 k2 k3 k4 k5*
Configures the metric constants used in the ISO-IGRP composite metric calculation. The arguments *qos, k1, k2, k3, k4, and k5* are metric constants. The argument qos must be 0. The default values for *k1, k2, k3, k4, and k5* are 1, 0, 1, 0, and 0 respectively.

[**no**] **net** *network-entity-title*
Configures a Network Entity Title (NET) for the routing process. The argument *network-entity-title* configures the domain, area, and system-id for an ISO-IGRP routing process or the area address and system-id for an IS-IS routing process.

[**no**] **redistribute isis** [*tag*]

Imports IS-IS routes into other protocols. The optional argument *tag* defines a meaningful name for a routing process. The default setting is **no redistribute isis**.

[**no**] **redistribute** *CLNS-routing-protocol* [*tag*]

Specifies the dynamic routes to be redistributed into the IS-IS or ISO-IGRP domain. The argument *CLNS-routing-protocol* specifies the routing protocol that is redistributed into IS-IS or ISO-IGRP. When redistributing CLNS prefix routes, only an CLNS routing protocol name or the keyword **static** is allowed. Supported *CLNS-routing-protocol* keywords are **static**, **isis**, and **iso-igrp**. Static routes are always redistributed into IS-IS unless a NO REDISTRIBUTE STATIC command is executed. Redistribution only occurs for level-2 routing.

The optional argument *tag* is the name of a process.

The REDISTRIBUTE global configuration command causes the routes learned by the routing process *tag* to be advertised into IS-IS or ISO-IGRP.

[**no**] **redistribute static** [**clns**]

Causes the router to inject any static CLNS prefix routes into the domain. The **no**form stops redistribution.

[**no**] **timers basic** *update holddown flush*

Configures ISO-IGRP timers. The argument *update* is the time, in seconds, between the sending of routing updates. The default value is 90 seconds. The argument *holddown* is the amount of time, in seconds, a system or area router is kept in holddown state. The default value is 145 seconds. The argument *flush* is the amount of time, in seconds, a route remains in the routing table after it has been determined that it is not reachable. The default value is 135 seconds.

Interface Subcommand Summary

This section provides an alphabetical list of the ISO CLNS interface subcommands.

[**no**] **clns router iso-igrp** *tag* [**level 2**]

Enables the routing process on the interface. The argument *tag* is the tag defined for the routing process in the preceding ROUTER ISO-IGRP global configuration command. If you want this interface to advertise Level 2 information only, use the **level 2** keyword.

[**no**] **clns router isis** *tag*

Enables IS-IS routing for CLNS on a specific interface. Use the same text described earlier for the argument *tag* as specified in the global configuration command ROUTER ISIS. The NO ROUTER ISIS global configuration command with the appropriate area tag disables IS-IS routing for the system.

[**no**] **isis circuit-type** [**level-1** | **level-1-2** | **level-2-only**]
Configures the type of adjacency desired for this interface. If **level-1** is specified, at most a Level 1 adjacency may be established if there is at least one area address in common between this system and its neighbors. If **level-1-2** is specified, a Level 1 and 2 adjacency is established if the neighbor is configured as **level-1-2** and there is at least one area in common. If there is no area in common, a Level 2 adjacency is established. If **level-2-only** is specified, a Level 2 adjacency is established if and only if the neighbor is configured exclusively to be a Level 2 router. The default value for this command is LEVEL-1-2. The NO ISIS CIRCUIT-TYPE command resets the circuit type to Level l and Level 2.

[**no**] **isis csnp-interval** *seconds* [**level-1** | **level-2**]
Specifies the interval of time between transmission of Complete Sequence Number PDUs. The argument *seconds* is the length of time between transmissions.

[**no**] **isis hello-interval seconds** [**level-1** | **level-2**]
Specifies the length of time in seconds between hello packets the router sends on the interface. The argument *seconds* is an unsigned integer value.

[**no**] **isis metric** *default-metric* [**level-1** | **level-2**]
Configures the metric (or cost) for the specified interface. The default value for the [**level-1** | *default-metric* argument is ten. You can configure this metric for Level 1 and/or Level 2 routing. The NO ISIS METRIC command resets the *default-metric* value to ten. Specifying the **level-1** or **level-2** optional keywords resets the metric only for Level 1 or Level 2 routing, respectively.

[**no**] **isis password** *password* [**level-1** | **level-2**]
Configures the authentication password for an interface. Different passwords can be assigned for different routing levels using the optional **level-1** and **level-2** keyword arguments. By default authentication is disabled. The NO ISIS PASSWORD command disables authentication for IS-IS. Specification of the **level-1** or **level-2** optional keywords disables the password only for Level 1 or Level 2 routing, respectively.

[**no**] **isis priority** *value* [**level-1** | **level-2**]
Configures the priority to use for designated router election. Priorities can be configured for Level 1 and Level 2 individually. The default *value* is 64. The NO ISIS PRIORITY command resets priority to 64. The higher the number, the higher the priority. Specification of the **level-1** or **level-2** optional keywords resets priority only for Level 1 or Level 2 routing, respectively.

[**no**] **isis retransmit-interval** *seconds*
{Text}

Configures the number of seconds between retransmissions of IS-IS LSP for point-to-point links. The argument *seconds* the length of time between retransmissions.

EXEC Command Summary

This section describes the ISO CLNS EXEC command in this chapter.

[**no**] **which-route** {*NSAP-address* | *CLNS-name*}
Displays the routing table in which the specified CLNS destination is found. The argument *NSAP-address* is the specified CLNS destination network address. The argument *CLNS-name* is the destination host name. See Chapter 1 of the *DECbrouter 90T Configuration and Reference Volume 3* publication for a complete description of the CLNS HOST command.

3
Routing Novell IPX

This chapter describes the DECbrouter 90's implementation of the Novell IPX routing protocol. You will find these topics and tasks described in this chapter:

- Overview of addressing in Novell IPX networks
- Basic Novell IPX routing configuration steps
- Processes for configuring optional parameters
- Steps for developing and using Novell IPX access lists
- Examples of controlling Novell broadcast messages and using various filters

The DECbrouter 90's Implementation of Novell IPX

Novell IPX is a variation on Xerox Network Systems (XNS). One major difference between IPX and XNS is that they do not use the same Ethernet encapsulation format. A second difference is that IPX uses Novell's proprietary Service Advertisement Protocol (SAP) to advertise special network services. A file server is one instance of a service that is typically advertised.

The DECbrouter 90's implementation of Novell's IPX protocol provides all of the functionality of a Novell "External Bridge" (Novell refers to its router functionality as bridging). As a Novell External Bridge, a DECbrouter 90 connects Ethernets either directly or through high-speed serial lines (56 kbps to T1 speeds) or X.25. Novell workstations on any LAN, including those without file servers, connect to Novell file servers on any other LAN. Novell sells an X.25 and a T1 interface capability. At this time, the Digital X.25 and T1 support is not compatible with Novell. This means that the DECbrouter 90 or Cisco Systems routers must be used on both ends of T1 and X.25 circuits.

Novell Addresses

Novell node IDs are 48-bit quantities, represented by dotted triplets of four-digit hexadecimal numbers. A Novell router will have interfaces on more than one physical network (Ethernet, Token Ring, serial line, and so on). Physical networks are identified by 32-bit numbers written in hexadecimal. These network numbers must be unique throughout a Novell internet. Since both the network number and the host address are needed to deliver traffic to a host, addresses are usually given as network numbers, followed by host addresses, separated with dots. An example would be:

```
4a.0000.0c00.23fe
```

Here, the network number is 4a, and the host address is 0000.0c00.23fe.

Note

All numbers in the address, including the network number 4a, are expressed in hexadecimal numbers.

Configuring Novell Routing

There are only two commands required to enable Novell IPX routing:

1. Enable routing using the global configuration command NOVELL ROUTING.
2. Assign Novell routing to a specific interface using the interface subcommand NOVELL NETWORK.

All other configuration commands provide additional functionality or refinements. Each task is described in the following section. These descriptions are followed by applicable EXEC commands for monitoring and debugging Novell networks. Summaries of global configuration commands and interface subcommands described here appear at the end of this chapter.

Novell Configuration Restrictions

An interface takes as its Novell host address the hardware MAC address currently assigned to the interface. If the MAC address is later changed to some other value, the Novell node address automatically changes to the new address. Of course, connectivity will be lost for awhile because of this change.

An optional address argument to the NOVELL ROUTING configuration command (see the following section) establishes the default Novell node address. This address is used as the Novell node address for any non-LAN interface, such as serial links.

Enabling Novell Routing

To enable or disable Novell routing, use the NOVELL ROUTING global configuration command. The full command syntax of this command follows.

novell routing [*host-address*]
no novell routing

The argument *host-address* is optional. If you do not specify an address, the MAC address of the Ethernet interface is used. If there are no satisfactory interfaces present, you must specify the host address using the optional *host-address* argument. The address must not be multicast. The NOVELL ROUTING command enables Novell RIP routing and SAP services. Novell network numbers must still be assigned to the appropriate interfaces with the NOVELL NETWORK subcommand.

Use the NO NOVELL ROUTING command to disable Novell IPX routing.

Enabling Novell on an Interface

To enable Novell routing on a particular interface, use the NOVELL NETWORK interface subcommand. The full syntax of this command follows.

novell network *number*
no novell network *number*

The argument *number* is the number of the Novell network to which that interface is attached. Novell packets received on an interface that does not have a Novell network number are ignored. Use the NO NOVELL NETWORK command with the network number to disable Novell on the interface.

Example

```
novell network 2f
```

Repairing Corrupted Network Numbers

In some early implementations of Novell client software, it was possible for the client's network number to be corrupted. The NOVELL SOURCE-NETWORK-UPDATE interface subcommand repairs corrupted network numbers by setting the source network field of any packet with a hop count of zero to the local network number. The full syntax of this command follows.

novell source-network-update
no novell source-network-update

The route cache must be disabled or this command will not work; use the NO NOVELL ROUTE-CACHE command.

Note

This command will interfere with the proper working of OS/2 Requestors. Do not use this command in a network where OS/2 Requestors are present.

Example

```
novell network 106A
novell source-network-update
no novell route-cache
```

Novell Encapsulation

Novell uses several different data formats on Ethernets. Use the NOVELL ENCAPSULATION interface subcommand to select which data format or encapsulation is used on an Ethernet interface.

novell encapsulation *keyword*

The default keyword argument is **novell-ether**, which specifies Novell IPX over Ethernet using Novell's variant of IEEE 802.2 encapsulation. Novell refers to this as "Frame Type Ethernet_802.3" in its documentation. The keyword **arpa** is used when the Novell systems must communicate with other vendors' systems, such as DEC VAX/VMS. Novell refers to this encapsulation as "Frame Type Ethernet _II." In this case, Ethernet-style encapsulation is used with a protocol type of 8137.

Note

Some Novell nodes do not recognize token ring packets with the source-route bridging RIF field set. You can work around this Novell discrepancy by using the NO MULTIRING interface subcommand on token ring interfaces that are used for Novell IPX routing.

Configuring Static Routes

Static routes for a Novell network can be specified with the NOVELL ROUTE global configuration command. The full syntax of the command follows.

novell route *network network.address*
no novell route *network network.address*

The NOVELL ROUTE command causes packets received for the specified network to be forwarded to the specified router, whether or not that router is sending out dynamic routing.

Use the NO NOVELL ROUTE command with the appropriate arguments to remove the route.

Example

If the router that handled traffic for network 5e had the address 3abc.0000.0c00.1ac9, you would enter this command:

```
novell route 5e 3abc.0000.0c00.1ac9
```

Note

Be careful when assigning static routes. When links associated with static routes are lost, traffic may stop being forwarded, although alternative paths are available.

Configuring Static SAP

Static Novell SAP table entries can be specified with the novell sap global configuration command. The full syntax of the command follows.

novell sap *service-type name network.address socket hop-count*
no novell sap *service-type name* [*network.address*] [*socket*] [*hop-count*]

The NOVELL SAP command provides a static entry in the SAP table for the associated service. Static SAP assignments always override any identical dynamically learned service-type/service-pair entry in the SAP table, regardless of hp count. For the NO NOVELL SAP command, only the *service-type* and *name* are required.

- *service-type* is the SAP service-type number
- *name* is the server name that provides the service
- *net.address* is the network and host address of the server
- *socket* is the socket number for this service

- *hop-count* is the number of hops away that the server is

If a dynamic route that is associated with a static SAP is lost or deleted, the static SAP will not be announced until the route is relearned.

Example

The following example establishes static SAP entries:

```
   novell sap 107 MAILSERV 160.0000.0c01.2b72 8104 1
   novell sap 4 FILESERV 165.0000.0c01.3d1b 451 1
   novell sap 143 OPTICALDISK A1.0000.0c01.1234 8170 2

   no route to A1, OPTICALDISK won't be announced until route is learned
```

In this example, the route to `OPTICALDISK` is not yet learned, and the system displays an informational message. The `OPTICALDISK` service will not be announced in the regular SAP updates until the router learns the route either by means of a RIP update from a neighbor or a static route command.

The output from a SHOW NOVELL SERVERS command at this point displays the following:

```
  test# show novell servers

 Total Novell Servers: 4

 Type   Name                         Net    Address      Port Hops Interface
    4  MAGNOLIA                      42.0000.0000.0001::0451  1  Ethernet2
    4  FILESERV                     165.0000.0c01.3d1b::0451  1  Ethernet0
  107  MAILSERV                     160.0000.0c01.2b72::8104  1  Ethernet2
  143  OPTICALDISK                  A1.0000.0c01.1234::8170  2
```

Notice that the server `OPTICALDISK` has no interface specification. This is because a route to network A1 has not yet been learned. After the route is added statically or heard via a RIP update, the router will learn the route and announce it by means of SAP updates on all interfaces except the one to the destination.

In addition to adding the route statically, the router can learn the route to network A1 by means of RIP updates from another router. If the route is learned statically, it can take up to one routing update-time to place the interface specification in the SAP table. The router will begin announcing the `OPTICALDISK` service in its next regular SAP update once the interface specification is in the SAP table.

Note

Be careful when assigning static SAPs. When the server associated with the static SAP is down, the static SAP entry will continue to be announced in SAP updates for as long as a route to the destination network exists. In addition, if the route to the destination was learned by means of a static route and the link associated with that link is lost, traffic will stop being forwarded to the static server even though an alternative path might be available.

Setting Maximum Paths

To set the maximum number of multiple paths that the router will remember and use, use the NOVELL MAXIMUM-PATHS global configuration command. The command was designed to increase throughput by using multiple paths. It remembers higher bandwidth routes in preference to lower bandwidth routes. The full syntax of the command follows.

novell maximum-paths *paths*
no novell maximum-paths

The argument *paths* is the number of paths to be remembered. For a given destination, multiple paths of equal cost will be remembered. The default value for *paths* is 1. Output will be determined in round-robin fashion over these multiple paths at the packet level.

The NO NOVELL MAXIMUM-PATHS command restores the default.

The EXEC command SHOW NOVELL ROUTES displays these additional routes and the maximum path value.

Setting Novell Update Timers

To allow the Novell routing update timers to be set on a per-interface basis, use the NOVELL UPDATE-TIME interface subcommand. Full syntax follows.

novell update-time *seconds*
no novell update-time

Internal Novell timers are affected by the value set for the *seconds* argument, as follows:

- Novell routes are marked invalid if no routing updates are heard within three times the value of the update timer and are advertised with a metric of infinity.
- Novell routes are removed from the routing table if no routing updates are heard within four times the value of the update timer.
- The default value for the **update-time** *seconds* argument is 60.
- The granularity of the update timer is determined by the lowest value defined.
- The minimum is ten seconds.

The NO NOVELL UPDATE-TIME command restores the default of 60 seconds.

Example

In this example, the granularity would be 20 because that is the lowest value specified for that protocol.

```
interface serial 0
novell update-time 40
interface ethernet 0
novell update-time 20
interface serial 1
novell update-time 25
```

Note

Be careful when using this command. It can be used only in an environment with DECbrouter 90s and Cisco routers, and all timers should be the same for routers connected to the same network segment.

The EXEC command SHOW NOVELL INTERFACE displays the value of these timers.

Filtering Novell Packets

The DECbrouter 90's implementation of the Novell IPX software provides three types of filtering:

- Access lists, or packet filtering, controls whether or not packets are sent out the filtered interface.
- Routing update filtering controls to which Novell IPX networks the router advertises routes to, depending on the type used.
- SAP filtering controls the services the router knows about and which services the router advertises.

In setting up these packet filters, take care not to set up filtering conditions that result in packets getting lost. This can happen, for example, when the software is configured to advertise services on a network with access lists configured to deny these packets, or when a network is configured to advertise services on a network that is unreachable because routing updates are filtered out by routing update filtering.

Keep these pitfalls in mind while configuring the filter types. Each is discussed in the following sections.

Configuring Novell IPX Access Lists

Simple Novell IPX access lists are numbered from 800 to 899 and filter on the source and destination addresses only.

The command syntax for standard Novell IPX access lists is lengthy. For typographic reasons, the command is shown here on multiple lines; it must be on a single line when given as a configuration command. The full command syntax for the Novell ACCESS-LIST global configuration command follows.

access-list *number* {**deny** | **permit**} *novell-source-network*[.*source-address* [*source-mask*]] *novell-destination-network* [.*destination-address* [*destination-mask*]]
no access-list *number*

The only required argument for standard Novell IPX access lists is the Novell IPX source network. The rest of the parameters are optional except that the source and/or destination address masks are present only if the corresponding source and/or destination address was entered.

Use the NO ACCESS-LIST command with the appropriate access list number to remove the access list.

Example

The following example denies access from source network -1 (all Novell IPX networks) to destination network 2.

```
access-list 800 deny -1 2
```

The following example denies access from Novell IPX source address 0000.0c00.1111.

```
access-list 800 deny 1.0000.0c00.1111
```

The following example denies access from all nodes on network 1 that have a source address beginning with 0000.0c.

```
access-list 800 deny 1.0000.0c00.1111 0000.00ff.ffff
```

The following example denies access from source address 1111.1111.1111 on network 1 to destination address 2222.2222.2222 on network 2.

```
access-list 800
 deny 1.1111.1111.1111 0000.0000.0000 2.2222.2222.2222 0000.0000.0000
```

Configuring Extended Novell Access Lists

Extended Novell IPX access lists filter on protocol information as well; numbers for the extended lists range from 900 to 999. The command syntax for extended Novell IPX access lists is again rather lengthy as a configuration command and must be typed on one line, even though it is shown here on multiple lines.

> **access-list** *number* {**deny** | **permit**} *novell-protocol source-network.* [*source-address* [*source-mask*]] *source-socket destination-network.* [*destination-address* [*destination-mask*]] *destination-socket*
> **no access-list** *number*

The source and destination addresses and masks are optional. The protocol number *novell-protocol* is the only required parameter. A network number of -1 matches all networks; a socket number of 0 matches all sockets.

Use the NO ACCESS-LIST command with the appropriate access list number to remove the access list.

Example

The following example denies access to protocol 1 from source network 1, source socket 1234 to destination network 2, destination socket 1234:

```
access-list 900 deny 1 1 1234 2 1234
```

The following example illustrates the use of all possible parameters:

```
access-list 900 deny 1 1.1111.1111.1111 0000.0000.0000 1234
  2.2222.2222.2222 0000.0000.0000 1234
```

Filtering Outgoing Traffic

The Novell IPX access list group number is assigned with the novell access-group interface subcommand. The syntax for this command follows.

novell access-group *number*
no novell access-group *number*

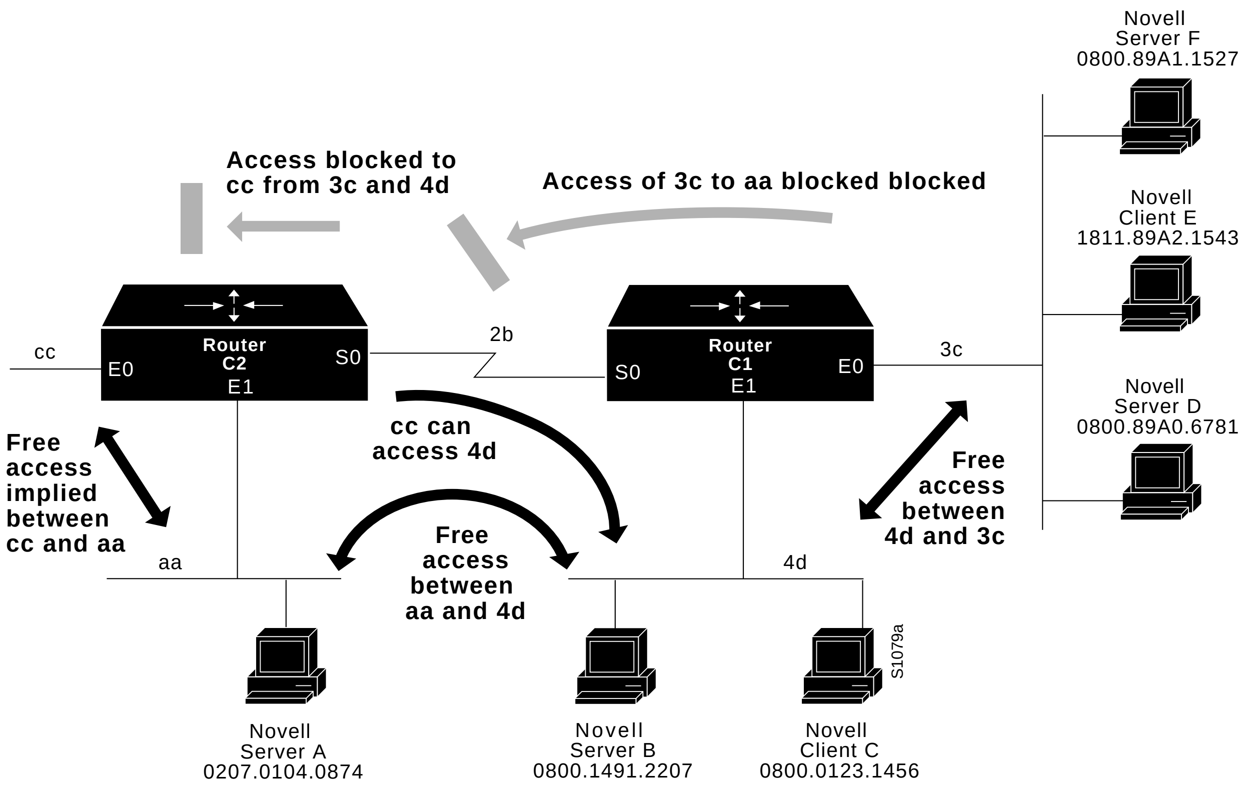
The argument *number* refers to the appropriate Novell access list number. All outgoing packets forwarded through the interface will be filtered by this access list.

Use the NO NOVELL ACCESS-GROUP command with the appropriate group number to remove the access list group number from the interface.

Example of Controlling Traffic with Access Lists

Using access lists to manage traffic routing can be a powerful tool in overall network control. However, it requires a certain amount of planning and the appropriate application of several related commands. Figure 3–1 illustrates a network featuring two Digital routers connecting a number of network segments. For the sake of clarity and simplicity, the routers shown in the examples have two Ethernet ports (E0 and E1), even though the DECbrouter 90 family supports only one Ethernet port.

Figure 3–1 Multiple Novell Servers Requiring Access Control



For the purposes of illustrating access control, the network in Figure 3–1 has the following specific requirements:

- Resources on networks 3c and 4d are to be allowed free access to each other through router C1.
- Resources on network 4d are to be allowed access to resources on network aa.
- Resources on network segment 3c are not allowed access resources on aa.
- Resources on network aa are not allowed to access resources on 3c.
- All networks can access resources on segment 4d.

The configuration for this environment can be defined using simple access lists as illustrated in the following example. In this example, the global configuration command ACCESS-LIST and interface subcommands NOVELL NETWORK and NOVELL ACCESS-GROUP are applied to router/bridge C1 in Figure 3–1.

Note

The commands in configuration files are executed in a sequential, first match, top down basis. Plan the placement of command lines carefully, especially when specifying and applying access lists. Access lists are always applied to the *transmitting* interface.

Example 1

This first example is applied to router C1, permitting resources on network 4d to access resources on network aa. It implicitly denies all other traffic.

```
access-list 800 permit 4d aa
```

If you want to explicitly deny all other traffic, add the following line:

```
access-list 800 deny -1 -1
```

Example 2

This example assigns network number 2b to the first serial interface, then applies access group 800 and permit resources on network 4d to access resources on network 3c. Again, you explicitly deny all other traffic.

```
interface serial 0
novell network 2b
novell access-group 800
access-list 800 permit 4d 3a
access-list 801 deny -1 -1
```

Example 3

This example assigns a network number and access group 801 to interface Ethernet 1 and applies a network number to the second Ethernet address. There are no explicit permissions or denials for interface Ethernet 1.

```
access-list 801 permit 4d 3c
interface ethernet 0
novell network 3c
novell access-group 801
interface ethernet 1
novell network 4d
```

Note

This configuration example does not address access control for segment cc on router C1. However, given no other restrictions or specific permissions, it implies that resources on cc can access resources on 4d, but cannot access resources on 3c. In addition, resources on 4d cannot access resources on network cc.

Filtering Novell Routing Updates

This section describes the filtering commands that use access lists to control which routing information is accepted or passed on within Novell networks. The commands filter incoming traffic, outgoing routing updates, and specific routers.

Each access list entry contains only one address parameter. How this address is interpreted is defined by the command that will use the list.

As with all other access lists, an implicit *deny everything* is defined at the end of the list. If this is not desired, an explicit *permit everything* definition must be included at the end of the list.

Each filter type is described in the following sections.

Establishing Input Filters

To control which networks are added to the routing table, use this interface subcommand:

novell input-network-filter *access-list-number*

The argument *access-list-number* is the access list number specified in the novell access-list command.

Example

In the following example, access list 876 controls which networks are added to the routing table when Novell routing updates are received. The address in the access list is a source network.

```
access-list 876 permit 1b
interface ethernet 1
novell input-network-filter 876
```

This configuration causes network 1b to be the only network that is accepted from updates received on the defined Ethernet interface.

Establishing Output Filters

To control the list of networks that are sent in routing updates, use the interface subcommand:

novell output-network-filter *access-list-number*

The argument *access-list-number* is the access list number specified in the novell access-list command.

Example

In the following example, access list 896 controls which networks are sent out in routing updates. The address parameter is the desired network.

```
access-list 896 permit 2b
interface serial 1
novell output-network-filter 896
```

This configuration causes network 2b to be the only network advertised in Novell routing updates sent on the defined serial interface.

Establishing Router Filters

To control the list of routers from which data will be accepted, use the following interface subcommand:

novell router-filter *access-list-number*

The argument *access-list-number* is the access list number specified in the novell access-list command.

Example

In this example, access list 866 controls from which router data will be accepted. In this case, the address parameter is the address of the router.

```
access-list 866 permit 3c.0000.000c0.047d
interface serial 0
novell router-filter 866
```

Information from a disallowed router is ignored.

Building SAP Filters

A common source of traffic on Novell networks is the SAP-based messages generated by Novell servers and routers as they broadcast their available capabilities. Control of SAP messages can be established with the DECbrouter 90 using several facilities. Access lists and SAP filters combine to allow you to control how SAP messages from network segments or specific servers are routed among Novell networks.

Defining Access Lists for SAP Filtering

To define an access list for filtering SAP requests, use this variation of the ACCESS-LIST command:

access-list *number* {**permit** | **deny**} *network.* [*address*] [*service-type*]

The argument *number* is the SAP access list, which is a decimal number in the range 1000 to 1099.

Enter the keyword **permit** or **deny** to establish the type of access desired. Permit or deny access is based on the data provided.

The argument *network* is a hexadecimal Novell network number; 0 defines the local network, -1 defines all networks.

The optional *address* argument is a Novell node address.

The *service-type* argument defines the service type to filter; 0 is all services. Service types are entered in hexadecimal. Examples of the service types that can be entered are listed in Table 3–1.

Table 3–1 Sample Novell SAP Services

Description	Service Type
Unknown	0
User	1
User Group	2
Print Queue	3
File Server	4
Job Server	5
Gateway	6
Print Server	7
Archive Queue	8
Archive Server	9
Job Queue	A
Administration	B
Remote Bridge Server	24
Advertising Printer Server	47
Wildcard	Blank (no entry)

Example—Service Type Specification

```
! Deny access from all nets for service 4:
access-list 1001 deny -1 4
! Permit access from all nets to all other services:
access-list 1001 permit -1
```

Note

In the preceding example, companion interface specifications must be defined for this access list to be applied. Once applied to an interface, this access list definition blocks all access to service type 4 (file service) on directly attached network by resources on other Novell networks. The second specification explicitly allows access to all other available services on the interface.

Configuring Novell SAP Filters

Use these commands to filter Novell SAP messages:

novell input-sap-filter *access-list-number*
novell output-sap-filter *access-list-number*
novell router-sap-filter *access-list-number*

These commands take a SAP Novell access list number as their input. The range for SAP lists is 1000 to 1099.

Follow these guidelines to use SAP filtering:

- When the **novell input-sap-filter** list is enabled, use the list to determine the services that will be accepted.
- When the **novell output-sap-filter** list is enabled, use the list to determine the services that will be included in SAP updates from Digital routers.
- When the **novell router-sap-filter** list is enabled, use the list to determine the router from which the DECbrouter 90 will hear SAP messages, and the service type.

Example SAP Input Filter

Input SAP filters are applied prior to the router accepting information about a service. In the example that follows, Router C1, illustrated in Figure 3–2, will not accept and, consequently not advertise, any information about Novell server F. However, C1 will accept information about all other servers on the network 3c. Router C2 will receive information about servers D and B in this example.

Example

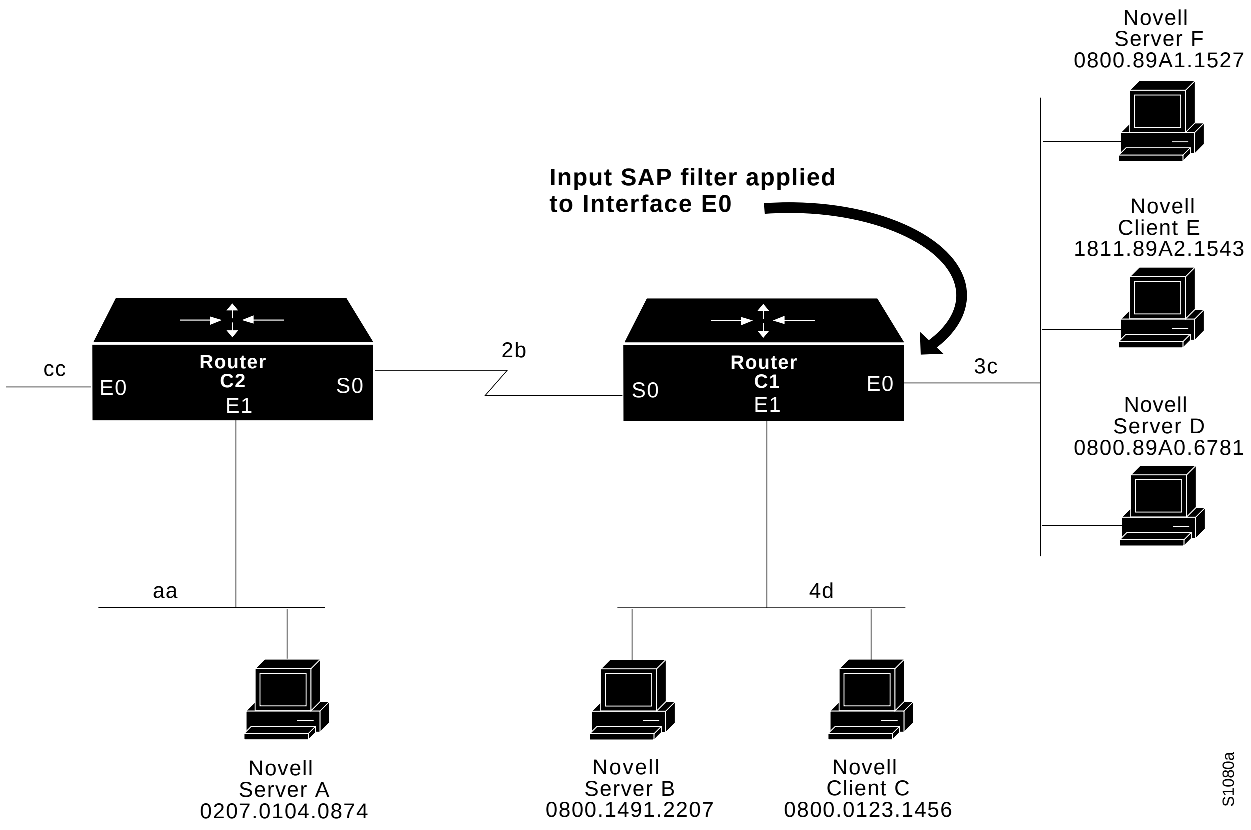
This example configures router C1. The first line denies server F. It accepts all other servers.

```
access-list 1000 deny 3c.0800.89a1.1527
access-list 1000 permit -1
interface ethernet 0
novell network 3c
novell input-sap-filter 1000
interface ethernet 1
novell network 4d
interface serial 0
novell network 2b
```

Note

NetWare 386 servers use an internal network and node number as their address for access list commands (the first configuration command in this example).

Figure 3–2 SAP Input Filter



Example SAP Output Filter

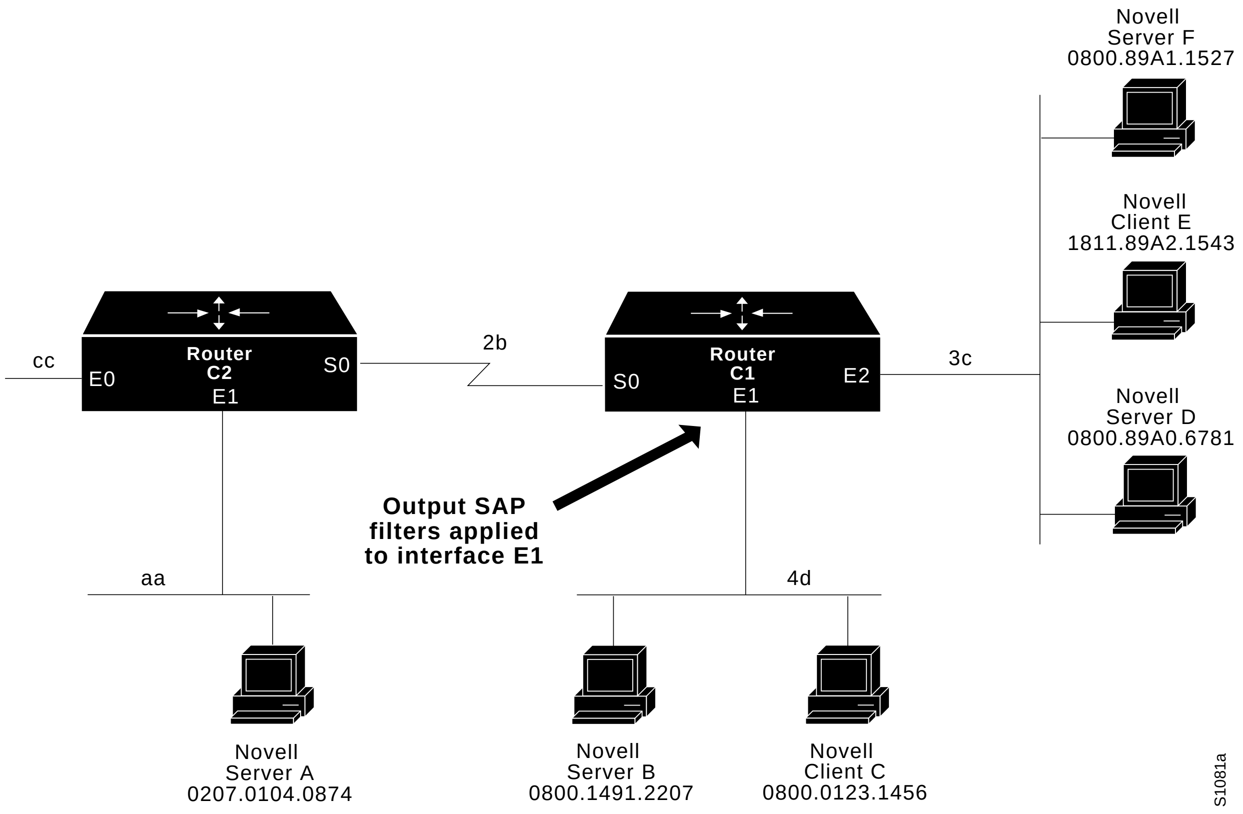
Output SAP filters are applied prior to the router sending information out a specific interface. In the example that follows, Router C1 (illustrated in Figure 3–3) is prevented from advertising information about Novell server A out interface Ethernet 1, but can advertise server A on network 3c.

Example

The following example refers to router C1. The first line denies server A. All other servers are permitted.

```
access-list 1000 deny aa.0207.0104.0874
access-list 1000 permit -1
interface ethernet 0
novell net 3c
interface ethernet 1
novell network 4d
novell output-sap-filter 1000
interface serial 0
novell network 2b
```

Figure 3–3 SAP Output Filter



Novell Broadcast Helper Facilities

The router's helper facilities provide a flexible set of tools to help you manage Novell network broadcast traffic. Several configuration options allow network administrators to tailor the way broadcasts generated by Novell clients are forwarded through a network.

If Novell clients and servers are attached to the same network segment, the function of blocking broadcasts is acceptable and often desired, as it helps reduce unwanted traffic among networks. However, when clients must broadcast through a router to a remotely located server, several modifications to the system configuration are required. To make these modifications, use the Novell helper functions.

The router supports flooding. *Flooding*, as the name suggests, forwards broadcasts to all networks.

The key to controlling Novell broadcasts (rather than simply blocking them) rests with the use of several commands specific to DECbrouter 90's Novell IPX routing implementation follows.

- **novell helper-address**—Explicitly specifies the address of a target Novell server (or network) to which broadcast packets are sent (applied to a specific interface).
- **novell helper-list**—Assigns access lists to interfaces to control broadcast traffic (applied to a specific interface).
- **access-list**—A general traffic-controlling tool that can be tailored to help manage broadcast traffic in a Novell network environment.

Defining a Helper List

The NOVELL HELPER-LIST and NOVELL HELPER-ADDRESS interface subcommands are defined briefly in this section and the one that follows, while the global configuration command ACCESS-LIST is described in a preceding section. Following the helper facility definitions, several typical applications illustrate how to use the helper and access list mechanisms together within the context of Novell-based internetworking environments.

The NOVELL HELPER-LIST interface subcommand specifies that only those packets that pass the specified Novell access list are forwarded to a remote Novell server. (The only exception to this rule is that all network (*all-nets*) flooded broadcasts (discussed in the next section) and NetBIOS are always forwarded, regardless of how you set the helper-list command.) The syntax for this command is as follows:

novell helper-list *access-list-number*

The argument *access-list-number* specifies the access list. The network numbers in that list are expressed in hexadecimal values.

Note

Since the destination address of a broadcast is by definition the broadcast address, this is only useful for filtering based on the source of the broadcast packet. This can be used to prevent nodes from discovering services they should not use.

Specifying Target Novell Servers

To forward broadcast packets that match the access list specified by the NOVELL HELPER-LIST subcommand, use the NOVELL HELPER-ADDRESS interface subcommand:

novell helper-address *net.host*

This subcommand causes all-nets broadcasts to be forwarded to *net.host*. The argument *net.host* is a dotted combination of the network and host addresses as explained in the NOVELL ROUTE subcommand.

Multiple NOVELL HELPER-ADDRESS commands can be entered on each interface, which will cause broadcast packets to be forwarded to each specified helper address.

Incoming unrecognized broadcast packets that match the access list will be forwarded on to the address specified by the argument net.host.

The router supports all-nets flooding. To configure the all-nets broadcast flooding, define the Novell helper address for an interface as follows:

```
-1.FFFF.FFFF.FFFF
```

On systems configured for Novell routing, this helper address will be displayed as:

```
FFFFFFFF:FFFF.FFFF.FFFF
```

Note

Although care has been taken to keep traffic to a minimum, some duplicates will be unavoidable. Under certain conditions (where loops exist) flooding can propagate bursts of excess traffic that will eventually age out when the hop count reaches its limit. Use the broadcast address carefully and only when necessary.

Using Helper Facilities to Control Broadcasts

Use of the **helper-list** and **helper-address** tools is best illustrated with examples. The following illustrations and accompanying descriptions outline the application of access lists, helper lists, and helper addresses to forward traffic to a specific network or to a node, and to flood broadcast messages on all attached links.

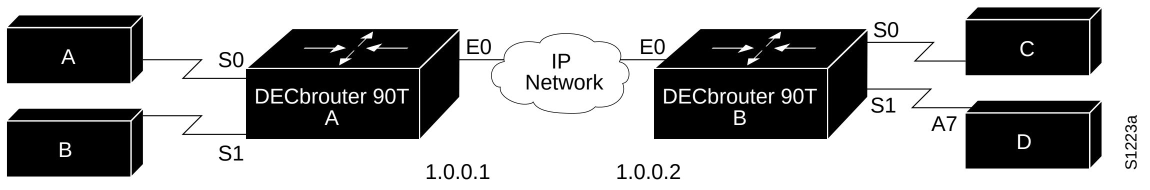
Forwarding to an Address

You can direct broadcasts to a specific network or host (node) on a segment. The following examples illustrate both these forwarding options.

Figure 3–4 shows three routers (C1,C2, and C3) connected to several Ethernets. In this environment, all Novell clients are attached to segment aa, while all servers are attached to segment bb. In controlling broadcasts, observe the conditions that follow.

- Only type 10 broadcasts are to be forwarded.
- The Novell clients on network aa are to be allowed to broadcast to any server on network bb.
- All-nets broadcasts are always flooded.

Figure 3–4 Novell Clients Requiring Server Access Through a Digital Router



Interfaces C2 and C3 do not require application of any specific permissions to meet the conditions for this example, because broadcasts are by default blocked by the router.

Example

This example configures the router C1 shown in Figure 3–4. The first line permits traffic of type 10 from network aa. Then the interface and network commands configure a specific interface. The HELPER-ADDRESS command permits broadcast forwarding from network aa to network bb. The last line forwards type 10 broadcasts from network aa to network bb.

```
access-list 900 permit 10 aa
interface ethernet 0
novell network aa
novell helper-address bb.ffff.ffff.ffff
novell helper-list 900
```

Any downstream network that is cascaded beyond network aa (for example, some arbitrary network aa1) will not be able to broadcast to network bb through router C1, unless the routers partitioning networks aa and aa1 are configured to forward broadcasts with a series of configuration entries analogous to the example provided for Figure 3–4. These entries must be applied to the input interface and be set to forward broadcasts between directly connected networks. In this way, traffic can be passed along in a directed manner from network to network.

Example

The following example, which uses multiple helper addresses, forwards broadcasts from network aa to network bb and network dd, but not to network cc as shown in Figure 3–4:

```
interface ethernet 0
novell network aa
novell helper-address bb.ffff.ffff.ffff
novell helper-address dd.ffff.ffff.ffff
```

Example

The following example rewrites the novell helper-address command line to direct broadcasts to server A in Figure 3–4:

```
novell helper-address bb.00b4.23cd.110a
! Permits node-specific broadcast forwarding to
! Server A at address 00b4.23cd.110a on network bb
```

Forwarding to All Networks

In some networks, it may be necessary to allow client nodes to broadcast to servers on multiple networks. If you configure your router to forward broadcasts to all attached networks, you are flooding the interfaces. To support this requirement, use the flooding address (-1.ffff.ffff.ffff) in your NOVELL HELPER-ADDRESS interface subcommand specifications.

Example

In this example, the first line permits traffic of type 10 to network 2b1. Then the Ethernet interface is configured with a network number. The helper address is defined and the helper list limits forwarding to type 10 traffic.

```
access-list 901 permit 10 2b1
interface ethernet 0
novell network 2b1
novell helper-address -1.ffff.ffff.ffff
novell helper-list 901
interface serial 0
novell network 3c2
interface serial 1
novell network 4a1
```

In this example, type 10 broadcasts from network 2b1 are forwarded to all directly connected networks. All other broadcasts are blocked. If all broadcasts are to be permitted, delete the **novell helper-list** entry.

Enabling Novell Fast Switching

Novell fast switching allows higher throughput by switching the packet using a cache created by previous transit packets. Fast switching also provides load sharing on a per-packet basis.

Use the NOVELL ROUTE-CACHE interface subcommand to enable fast switching. The full syntax for this command follows.

novell route-cache
no novell route-cache

When Novell routing is enabled, by default, Novell fast switching is enabled on the appropriate interface. Use the NO NOVELL ROUTE-CACHE command to disable fast switching.

Restricting SAP Updates

To configure less frequent SAP updates over slow links, use the interface subcommand NOVELL SAP-INTERVAL. This command has the following syntax:

novell sap-interval *interval*

Use the *interval* argument to set the interval between SAP updates. If *interval* is zero, periodic updates are not sent. A message is sent only when the server first appears and when it goes down. The default value for the argument *interval* is one minute. This is the value used by the Novell servers.

Example

In this example, SAP updates are sent (and expected) on interface serial 0 every five minutes.

```
interface serial 0
novell sap-interval 5
```

All Novell servers and routers on a particular network require the same SAP interval or they are likely to decide that a server is down, even though it is up. Since it is impossible to change this value on most PC-based servers, you should never change the interval for an Ethernet that has actual servers on it. This subcommand is most useful on limited bandwidth point-to-point links or X.25 interfaces, where one prefers to use as little bandwidth as possible for sending the SAP updates.

Note

Setting the interval to zero is especially dangerous. Routers that were inaccessible for any reason at the time of a server power-up or shutdown will miss that event, and will either fail to learn about the existence of new servers or will fail to detect the server shut down.

SAP Update Delays

Some slow Novell servers lose SAP updates because they cannot keep up the same brisk processing pace that the DECbrouter 90 can. The NOVELL OUTPUT-SAP-DELAY interface subcommand allows you to set a delay between packets in a multipacket SAP update, in effect forcing the DECbrouter 90 interface to pace its output to the slower-processing needs of the Novell servers. If your server is not slow and is not losing SAP updates, you can skip this configuration command. The full syntax of the command follows.

novell output-sap-delay *delay*
no novell output-sap-delay

The parameter *delay* is measured in milliseconds.

The **no novell output-sap-delay** disables the delay mechanism.

Example

```
novell network 106A
novell output-sap-delay 20
```

Novell Configuration Example

The following configuration commands enable Novell routing, defaulting the Novell host address to that of the first IEEE-conformance (no serial, for example) interface. Routing is then enabled on Ethernet 0 for Novell network 2abc.

Example

```
novell routing
interface ethernet 0
novell network 2abc
```

Note

The network numbers used must match the numbers used by Novell file servers on the same cable.

Monitoring the Novell IPX Network

Use the EXEC commands described in this section to obtain displays of activity on the Novell IPX network.

Displaying the Novell Cache Entries

Use the SHOW NOVELL CACHE command to display a list of fast-switching cache entries. Enter this command at the EXEC prompt:

show novell cache

Following is sample output:

```
Novell routing cache version is 9
Destination        Interface                  MAC Header
*1006A              Ethernet0           00000C0062E600000C003EB0064
```

In the sample display, valid entries are marked by an asterisk (*).

Displaying Novell Interface Parameters

Use the SHOW NOVELL INTERFACE command to display the Novell parameters that have been configured on the interfaces. Enter this command at the EXEC prompt:

show novell interface [*interface unit*]

An optional interface name can be specified with the *interface unit* arguments to see information for a specific interface. Following is sample output:

```
Ethernet 0 is up, line protocol is up
  Novell encapsulation is NOVELL-ETHER
  Novell address is 1006A.0000.0c00.62e6
  Outgoing access list is not set
  Novell SAP update interval is 1 minute(s)
  Novell Helper access list is not set
  SAP Input filter list is not set
  SAP Output filter list is not set
  SAP Router filter list is not set
  Input filter list is not set
  Output filter list is not set
  Router filter list is not set
  Update time is 30 seconds
  NOVELL Fast switching enabled
```

Displaying the Novell Routing Table

Use the SHOW NOVELL ROUTE EXEC command to display the Novell routing table. Enter this command at the EXEC prompt:

show novell route

Following is sample output:

```
Codes: R - RIP derived, C - connected, S - static, 2 learned routes

Maximum allowed path(s) are/is 1
R Net 1001 [1/1] via 1006.aa00.0400.6508,  94 sec, 0 uses, Ethernet0
R Net 1003 [1/1] via 1006.aa00.0400.6508,  94 sec, 0 uses, Ethernet0
C Net 13A is directly connected, 0 uses, Serial1 (down)
C Net 1006A is directly connected, 0 uses, Ethernet0
```

In the display, the leading character R indicates routes learned via RIP, C indicates connected entries, and S indicates statically defined entries.

Displaying Novell Servers

Use the SHOW NOVELL SERVERS EXEC command to list the servers discovered through SAP advertisements. Enter this command at the EXEC prompt:

show novell servers

Following is sample output:

```
Type Name Net Address       Port Hops Interface
4 SYSOP 2a.0206.00a2.41ec:0450  2 Ethernet0
4 MKTG 3c.0800.00a3.45ef:0450  2 Ethernet0
4 SERVICE 4d.0a44.008a.0220:0450  3 Serial0
4 FINANCE 1a.0080.a246.0001:0450  3 Serial1
```

For each known server in the network, the display lists its name, complete address, number of hops distant it is and the interface through which it can be accessed (through which it was discovered).

Displaying Novell Traffic

Use the SHOW NOVELL TRAFFIC EXEC command to display information on the number and type of Novell packets transmitted and received. Enter this command at the EXEC prompt:

show novell traffic

Following is sample output:

```
Rcvd:   68112 total, 0 format errors, 0 checksum errors, 0 bad hop count,
        68102 local destination, 0 multicast
Bcast:  68102 received, 43745 sent
Sent:   43745 generated, 0 forwarded
        0 encapsulation failed, 10 no route
SAP:    0 SAP requests, 0 SAP replies
        0 SAP advertisements received, 0 sent
Echo: Rcvd 0 requests, 0 replies
      Sent 0 requests, 0 replies
        0 unknown
```

The following notes apply to the less obvious statistics in this screen:

- `Format errors` are reported whenever a bad packet is detected (for example, a corrupted header).
- `Checksum errors` should not be reported, because IPX does not use a checksum.
- `Bad hop count` increments when a packets hop count exceeds 16.
- `Encapsulation failed` is registered when the router is unable to encapsulate a packet.
- The `unknown` counter increments when packets are encountered that the router is unable to forward (for example, misconfigured helper address, or no route available).

The Novell PING Command

To execute the PING command on a DECbrouter 90 configured for Novell routing, enter **novell** at the ping protocol prompt and the Novell routing address at the address parameter prompt. The defaults are enclosed in brackets at each prompt.

Here is a sample:

```
Protocol [ip]: novell
Target Novell Address: 1006A.0000.0c00.62e6
Repeat Count [5]:
Datagram Size [100]:
Timeout in seconds [2]:
Verbose [n]:
Type escape sequence to abort.
Sending 5 100-byte Novell echoes to 1006A,0000,0c00,62e6,
timeout is 2 seconds.
!!!!!!!
Success rate is 100%, round trip min/avg/max  = 1/2/4 ms.
```

Note

This command only works on DECbrouter 90 or Cisco routers. Novell devices will not respond to this command.

Debugging the Novell IPX Network

Use the commands described in this section to troubleshoot and monitor the Novell IPX network. For each DEBUG command, these is a corresponding UNDEBUG command that turns off message logging.

debug novell-packet

The DEBUG NOVELL-PACKET command outputs information about packets received, transmitted, and forwarded.

debug novell-routing

The DEBUG NOVELL-ROUTING command prints out information on Novell routing packets.

debug novell-sap

The DEBUG NOVELL-SAP command displays additional information about Novell Service Advertisement packets.

Novell IPX Global Configuration Command Summary

The following is an alphabetical list of the Novell IPX global configuration commands. These commands specify system-wide parameters for Novell IPX support.

[**no**] **access-list** *number* {*deny* | *permit*} *novell-protocol source-network.* [*source-address* [*source-mask*]] *source-socket destination-network.* [*destination-address* [*destination-mask*]] *destination-socket*

Specifies extended Novell IPX access lists. The source and destination addresses and masks are optional. The protocol number *novell-protocol* is the only required parameter. A network number of -1 matches all networks; a socket number of 0 matches all sockets. Extended Novell IPX access lists filter on protocol information as well; numbers for the extended lists range from 900 to 999. The **no** form of the command removes any access list in the current image with the specified number.

[**no**] **access-list** *number* {**deny** | **permit**} *novell-source-network* [*.source-address* [*source-mask*]] *novell-destination-network* [*.destination-address* [*destination-mask*]]

Specifies standard Novell IPX access lists. Standard Novell IPX access lists are numbered from 800 to 899 and filter on the source and destination addresses only. An access list command must be completely specified on a single line when given as a configuration command. The only required parameter for standard Novell IPX access lists is the Novell IPX source network. The rest of the parameters are optional except that the source and/or destination address masks are present only if the corresponding source and/or destination address was entered. The **no** form of the command removes any access list in the current image with the specified number.

[**no**] **access-list** *number* {**permit** | **deny**} *network*.[*address*] [*service-type*]

Defines an access list for filtering SAP requests. The argument *number* is a decimal number in the range of 1000 to 1099. Enter the keyword **permit** or **deny** to establish the type of access desired. Permit or deny access is based on the data provided. The argument *network* is a hexadecimal Novell network number; 0 defines the local network, -1 defines all networks. The optional *address* argument is a Novell host address. The *service-type argument* defines the service type to filter; 0 is all services. Service types are entered in hexadecimal.

[**no**] **novell input-sap-filter** *access-list-number*
[**no**] **novell output-sap-filter** *access-list-number*

[**no**] **novell router-sap-filter** *access-list-number*

Configure the router to filter the acceptable source of Novell SAP messages; the intended destination of SAP messages; or the specific router from which SAP filters will be accepted. These commands take a SAP Novell access list number as their input. The range for SAP lists is 1000 to 1099. The **no** forms of the commands remove the filters.

[**no**] **novell maximum-paths** *paths*

Sets the maximum number of multiple paths that the router will remember and use. The argument *paths* is the number of paths to be remembered. The **no** form of the command restores the default.

[**no**] **novell route** *network network.address*

Specifies or removes static routes for a Novell network. When specified, the command causes packets received for the specified network to be forwarded to the specified router, whether or not that router is sending out dynamic routing.

[**no**] **novell routing** [*host-address*]

Enables and disables Novell routing and Novell RIP routing and SAP services. You can also use this command to specify the system-wide host address to use with the optional argument *host-address*. If you do not specify an address, the MAC address of the first Ethernet interface is used. If there are no satisfactory interfaces present, you must specify the host address argument. The address must not be multicast. Assign Novell network numbers to the appropriate interfaces with the NOVELL NETWORK subcommand.

[**no**] **novell sap** *service-type name network.address socket hop-count*

The NOVELL SAP command provides a static entry in the SAP table for the associated service. Static SAP assignments always override any identical dynamically learned service-type/service-pair entry in the SAP table, regardless of hp count. For the NO NOVELL SAP command, only the *service-type* and *name* are required.

Novell IPX Interface Subcommand Summary

The following is an alphabetical list of the Novell IPX interface subcommands. These subcommands specify line-specific parameters for Novell IPX support. These subcommands must be preceded by an INTERFACE command.

[**no**] **novell access-group** *number*

Assigns or removes a Novell IPX access list group number to a specific interface. The argument *number* refers to the appropriate Novell access list number. All outgoing packets forwarded through the interface will be filtered by this access list.

novell encapsulation *keyword*

Selects which data format or encapsulation is used on an Ethernet interface. The default keyword argument is **novell-ether** which specifies Novell IPX over Ethernet using Novell's variant of IEEE 802.2 encapsulation. The keyword **arpa** is used when the Novell systems must communicate with other vendors' systems, such as Digital VAX/VMS. In this case, Ethernet-style encapsulation is used with a protocol type of 8137.

[**no**] **novell helper-address** *net.host*

Forwards broadcast packets that match the access list specified by the NOVELL HELPER-LIST subcommand. This subcommand causes all-nets broadcasts to be forwarded to *net.host*. The argument *net.host* is a dotted combination of the network and host addresses as explained in the NOVELL ROUTE subcommand. Incoming unrecognized broadcast packets that match the access list will be forwarded on the address specified by the argument *net.host*. This subcommand is useful for hosts which use a protocol other than SAP for advertising their availability.

[**no**] **novell helper-list** *access-list-number*

Specifies that only those packets that pass the specified Novell access list will be forwarded to the Novell helper host. The argument *access-list-number* specifies the access list. The network numbers in that list are expressed in hexadecimal values. The **no** form of the command disables the function.

[**no**] **novell input-network-filter** *access-list-number*

Explicitly specifies which networks are added to the Novell IPX routing table. The argument *access-list-number* is the access list number specified in the NOVELL ACCESS-LIST command. The no form of the command disables the function.

[**no**] **novell network** *number*

Enable and disables Novell routing on a particular interface. The argument *number* is the number of the Novell network to which that interface is attached. Novell packets received on an interface that do not have a Novell network number are ignored.

[**no**] **novell output-network-filter** *access-list-number*

Explicitly specifies the list of networks that are sent in routing updates. The argument *access-list-number* is the access list number specified in the

NOVELL ACCESS-LIST command. The **no** form of the command disables the function.

[**no**] **novell output-sap-delay** *delay*

Sets the interval, measured in milliseconds, between individual packets in a multipacket SAP update. The **no** form of the command disables the mechanism.

[**no**] **novell route-cache**

Enables and disables Novell fast switching. When routing is enabled, by default, Novell fast-switching is enabled on the appropriate interface. The **no** form of the command disables fast switching.

[**no**] **novell router-filter** *access-list-number*

Specifies or removes the list of routers from which data will be accepted. The argument *access-list-number* is the access list number specified in the NOVELL ACCESS-LIST command.

novell sap-interval *interval*

Configures less frequent SAP updates over slow links by setting the interval between SAP updates to the number of minutes specified by the *interval* argument. If *interval* is zero, periodic updates are not sent. A message is sent only when the server first appears and when it goes down. The default value for the argument *interval* is one minute. This is the value used by the Novell servers.

[**no**] **novell source-network-update**

Enables the interface to provide the current network number in place of the source network number of any packet that arrives with a hop count of zero. The no form of the command disables the function.

[**no**] **novell update-time** *seconds*

Allows the Novell routing update timers to be set on a per-interface basis.The **no** version of the command restores the default of 60 seconds.

4 Routing PUP

This chapter describes routing configuration using the Xerox PARC Universal Protocol (PUP). Monitoring and debugging commands are also described.

The DECbrouter 90's Implementation of PUP

PUP was developed at Xerox's Palo Alto Research Center in the mid-1970s. PUP originally ran on the experimental three-megabits per second (Mbps) Ethernet, the precursor to the IEEE 802.3 and Ethernet standards. The protocol is still used today by some Xerox workstations. The DECbrouter 90 routers support routing the PUP and provide a minimal set of PUP host functions, primarily the PUP Echo server and client. PUP packets can be routed over all DECbrouter 90-supported media. The DECbrouter 90 PUP implementation uses the PUP GWINFO routing protocol to maintain routing information across a PUP network.

PUP Addresses

PUP addresses are 16-bit quantities written as two octal numbers separated by a pound sign (#). The following is an example of a PUP address:

```
10#371
```

The most significant eight bits of the address constitute the PUP network number, and the least significant eight bits constitute the PUP host number. In the example case, the network number is 10, and the host number is 371 (both in octal).

Configuring PUP Routing

PUP routing is enabled and disabled with the PUP ROUTING global configuration command, which has this simple syntax:

pup routing
no pup routing

When you start the PUP router process, the router begins routing PUP packets and sending out PUP routing updates.

PUP routing is enabled on a per interface basis by the PUP ADDRESS interface subcommand. The syntax of this command follows:

pup address *address*
no pup address

The argument *address* is the desired PUP address of the interface.

PUP routing can be disabled per interface by using the NO PUP ADDRESS subcommand.

The default state is for PUP routing to be disabled.

PUP Miscellaneous Services

The DECbrouter 90 PUP support allows broadcasts to the PUP Miscellaneous Services socket to be forwarded to another network (the helper address) where the appropriate servers can be found. This function is available in either mode of routing operation. To set the helper address, use the PUP HELPER-ADDRESS interface subcommand, which has this syntax:

pup helper-address *address*
no pup helper-address

The argument *address* is the PUP address of the desired server in this format.

The PUP PING Command

The privileged EXEC command PING sends PUP Echo packets when the keyword **pup** is specified as the protocol. The ping menu also prompts you for a PUP protocol address; enter an address to begin the exchange.

Monitoring PUP

Use the EXEC commands described in this section to display statistics about the PUP network.

show pup arp

The command show pup arp displays PUP-specific ARP entries.

show pup route

The command SHOW PUP ROUTE displays routing entries obtained from the PUP routing protocol.

show pup traffic

The command SHOW PUP TRAFFIC displays statistical summaries of PUP packets when PUP routing is enabled.

Debugging PUP

Use the EXEC commands described in this section to display reports about activity on the PUP network. For each DEBUG command, there is a corresponding UNDEBUG command that turns message logging off.

debug pup-packet

The command DEBUG PUP-PACKET enables logging of PUP routing activity to the console terminal.

debug pup-routing

The command DEBUG PUP-ROUTING enables logging of PUP GWINFO routing exchanges to the console terminal.

5 Routing VINES

This chapter describes DECbrouter 90's implementation of the Banyan VINES routing protocol. It provides the following information:

- The DECbrouter 90's implementation of VINES
- Configuring routing in a VINES network, including special customizing procedures for such tasks as address resolution and filtering packets through the network
- Maintaining and monitoring the VINES network

The DECbrouter 90's Implementation of VINES

The Banyan VINES protocol is a networking system for personal computers. "VINES" stands for Virtual Network System. This proprietary protocol was developed by Banyan, and is derived from Xerox's XNS protocol. The DECbrouter 90's implementation of VINES has been designed in conjunction with Banyan. It is a separate effort from DECbrouter 90's XNS routing protocol support.

The DECbrouter 90's implementation of VINES provides routing of VINES packets on all media types. Although software automatically determines a metric value that it uses to route updates based on the delay set for the interface, the DECbrouter 90 software implementation allows you to customize the metric.

The DECbrouter 90's VINES software offers address resolution to respond to address requests and broadcast addresses propagation. MAC-level encapsulation is also available for Ethernet or IEEE 802.2. Name-to-address binding for VINES host names also is supported, as are access lists to filter packets to or from a specific network. The VINES implementation also includes EXEC-level commands for maintaining, monitoring, and debugging the VINES network.

Configuring VINES

There are only two commands required to enable VINES routing:

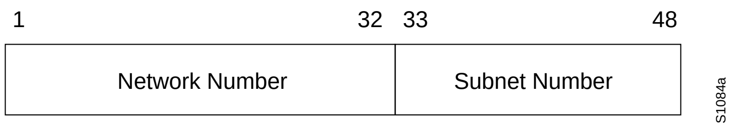
1. Use the global configuration command VINES ROUTING to enable routing.
2. Use the interface subcommand VINES METRIC to enable VINES processing on the interface.

All other configuration commands provide additional functionality or refinements. Each task is described in the following sections, which are followed by descriptions of the EXEC commands to maintain, monitor and debug the VINES network. Summaries of the global configuration commands and interface subcommands described in this chapter appear at the end of this chapter.

VINES Addressing

All VINES products automatically determine their addresses. The VINES specification guarantees interoperability. The network-level address consists of two numbers: a 32-bit network number and a 16-bit number that Banyan refers to as a *subnet number* but that actually serves as a host number. Figure 5–1 illustrates the address format.

Figure 5–1 Banyan VINES Network-Level Addresses



Address conflicts are impossible, because VINES servers use their Banyan-assigned unique key serial numbers as their network numbers and use a subnet number of one. Because the keys are unique, the server addresses are unique. VINES clients do not have addresses, as such. The clients use a modified version of the address of the first file server found on the physical network. The clients assume the server's network number and are assigned a subnet number by that server.

Using this address assignment scheme, it is likely that two clients on the same physical LAN will have different addresses. This means that the router must keep a cache of local neighbors, as well as of routing entries.

Banyan has assigned Digital a portion of the overall VINES network number space. This portion is the set of all numbers that begin with the first 11 bits (of the 32) of 0110 0000 000. This number set will appear in all Digital displays as a hexadecimal number beginning with 0x300 or 0x301. Routers attempt to automatically map themselves into the DECbrouter 90 number space based upon the first nonzero Ethernet address found. In the unlikely event that two routers map themselves to the same address, you can use the optional arguments to the VINES ROUTING command (see the section Configuring VINES Routing to override this selection.

The VINES Routing Table Protocol

Neighboring clients, servers, and routers are found by using the Routing Table Protocol (RTP). Every VINES client sends a broadcast Hello message at periodic intervals. This message indicates that the client is still operating and is reachable from the network. For VINES Version 3, this interval is 30 seconds. It has been increased to once every 90 seconds in the VINES Version 4 release. VINES servers send both Hello and update messages periodically. The update message is used to indicate which remote servers and their satellite clients can be reached by routing through that server. These messages are sent by VINES servers every 90 seconds. The DECbrouter 90 also broadcasts update messages once every 90 seconds; however, they do not send Hello messages, because they are redundant when updates are being sent.

Configuring VINES Routing

Use the global configuration command VINES ROUTING to enable VINES routing. The command has this syntax:

vines routing [*address*]
no vines routing

Specify the optional *address* argument only when you are *not* using an Ethernet interface, because otherwise the router automatically maps itself into the proper VINES address space. When the optional *address* argument is used, the routers' VINES address is automatically configured as the specified address.

Use the VINES METRIC interface subcommand to enable VINES processing on the defined interface. The command has this syntax:

vines metric [*number*]
no vines metric

The system automatically chooses a reasonable metric value that it uses in routing updates based upon the delay value set for the interface. These numbers are chosen to match as closely as possible to the numbers that a Banyan server would choose for the same type and speed of interface. Use the optional *number* argument to force the delay metric for the interface to a specific value. Banyan servers use delay metrics to compute timeouts when communicating with other hosts. Be careful when forcing metrics that you do not set this number too high or too low; doing so could disrupt the normal function of the Banyan servers. This number is generally inversely proportional to the speed of the interface. Some example numbers are found in Table 5–1.

Table 5–1 Interfaces and Example Delay Metric Values

Interface Type	Delay Metric Value
Ethernet	2
56 Kb Serial	45
9600 Serial	90

Example

These commands enable VINES routing on the Ethernet 0 interface:

```
!
vines routing
!
interface ethernet 0
vines metric 2
!
```

The VINES REDIRECT-INTERVAL interface subcommand determines how frequently a router sends an ICP redirect message on any given interface. The command has this syntax:

vines redirect-interval [*number*]
no vines redirect-interval

The optional *number* argument indicates, in seconds, how frequently the router should send redirect messages. If the number specified is zero, the router will never send redirects on the interface.

Example

To prevent redirect messages from ever being sent on the Ethernet 0 interface, use the following configuration:

```
!
vines routing
!
interface Ethernet 0
vines metric 2
vines redirect-interval 0
!
```

Configuring Address Resolution

VINES clients boot without any knowledge of network-level addresses and preferred servers. The first thing that a client does after initializing its hardware interface is send a broadcast message looking for available servers on the network to which it is attached. The client then waits for responses from the network. Once a message is received, the client sends an address assignment request to the first server that replied to the previous message. That server computes a new unique address based on its own network number and assigns the address to that client.

The address assignment interaction is accomplished by sending responses directly to the client's MAC addresses, because it does not have a network-level address yet. Table 5–2 depicts these events.

Table 5–2 Vines Address Assignment Method

Client	→	Broadcast:Are there any servers?
Client	←	Server2: Present
Client	←	Server1: Present
Client	←	Server3: Present
Client	→	Server2: Please assign me an address
Client	←	Server2: Your address is Server2:xxxx

The DECbrouter 90 normally does not respond to any of these messages. There is only one situation in which a response would be necessary; this is when a *stub* network (an Ethernet network with only one connection to a router) has only clients and no server. The router must be explicitly configured to provide address resolution functionality. By turning on the address resolution feature, the router will begin responding to address requests and assigning addresses. The router then acts as a network communication service provider for the client. There still must be a VINES file server somewhere on the network for the client to connect to for all other services. Use the command VINES ARP-ENABLE to enable this feature. The command has this syntax:

vines arp-enable
no vines arp-enable

The router generates a unique network number in a range assigned by Banyan, based on its VINES address.

A Banyan VINES network without a server needs to be configured with the VINES SERVERLESS interface subcommand. This command provides special processing for certain broadcast packets and certain packets directed at the router. It is necessary for proper functioning of the clients on a network without a server. This is especially important when two networks, one with a server and one without a server, are connected to the same router; in this situation you must use this command to build a path between the two networks. The command has this syntax:

vines serverless
no vines serverless

The default is **no vines serverless**.

Example

Use the following example commands when interface Serial 1 is a stub Ethernet that does not contain any VINES servers:

```
interface ethernet 0
vines metric 2
!
interface serial 1
vines metric 2
vines arp-enable
vines serverless
!
```

See the section VINES Configuration Examples for additional examples of this command.

Configuring VINES Encapsulation

Use the interface subcommand VINES ENCAPSULATION to set the MAC-level encapsulation used for VINES broadcasts on the defined interface. The command has this syntax:

vines encapsulation [**arpa** | **snap**]

You can choose from these encapsulation types:

- **arpa** for Ethernet lines
- **snap** for IEEE 802.2 media

The default encapsulation for Ethernet interfaces is **arpa**. All other media defaults to snap. As Banyan migrates to IEEE 802.2 encapsulation in future releases, the default will become **snap** for all media types.

Example

This example configures the IEEE 802.2 SNAP encapsulation.

```
vines routing
!
interface ethernet 0
vines metric 2
vines encapsulation snap
```

Please note that the VINES ENCAPSULATION command only affects broadcasts from the router. The router keeps track of which encapsulation is used by each of its neighbors and will use the same style of encapsulation when talking directly to a neighbor.

Configuring Name-to-Address Mappings

The DECbrouter 90 provides only static name-to-address bindings for the VINES protocol. (This is completely separate from Banyan's StreetTalk implementation.) Once entered, a VINES host name can be used anywhere that a VINES address can be used. This simplifies typing commands, because it is much easier to remember and type the name of a system than it is to remember and type its address. Names are entered by using the VINES HOST configuration command. The command has this syntax:

vines host *name address*
no vines host *name*

The argument *name* can be any length and sequence of characters separated by white space. The argument *address* consists of a VINES address in the form of network: subnet.

Example

These commands define four host names.

```
!
! Digital names
vines host FARSLAYER 60002A2D:0001
vines host DOOMGIVER 60000A83:0001
! VINES PS/2 server
vines host COINSPINNER 0027AF92:0001
! PC clone client
vines host STUFF 0027AF92:8001
!
```

To examine the cache of VINES name-to-address mappings, use the EXEC command SHOW VINES HOST.

The output appears as follows.

```
VINES Name Table
            FARSLAYER                      60002A2D:0001
            DOOMGIVER                      60000A83:0001
            COINSPINNER                    0027AF92:0001
            STUFF                          0027AF92:8001
```

See the section Monitoring the VINES Network for more information about this command.

Configuring VINES Access Lists

An access list is a list of VINES network numbers kept by the router to control access to or from a specific network for a number of services.

The DECbrouter 90 VINES access list provides network security by permitting or denying certain packets onto a network interface.

Use the global configuration command VINES ACCESS-LIST to create or delete a VINES access list. The command has one syntax for IP and another for other protocols.

> **vines access-list** *number* {**permit** | **deny**} **IP** *source-address source-mask dest-address/dest-mask*
> **vines access-list** *number* {**permit** | **deny**} *protocol source-address source-mask source-port/dest-address dest-mask dest-port*
> **no vines access-list** *number*

The argument *number* must be an integer between 1 and 100 for both variations of the command. The argument *protocol* can be an integer between 1 and 255 or one of these protocols:

- **SPP**—Sequence Packets Protocol
- **RTP**—Routing Table Protocol
- **ARP**—Address Resolution Protocol
- **IPC**—Interprocess Communications
- **ICP**—Internet Control Protocol

Source and destination addresses (*source-address and dest-address*) are VINES addresses written in the standard form.

Source and destination masks (*source-mask and dest-mask*) are also VINES addresses in the standard form, but they indicate the bits in the corresponding address that should be ignored.

Port numbers (*source-port and dest-port*) are integers in the range of 0 to 65535.

Use the interface subcommand VINES ACCESS-GROUP to assign an outgoing VINES access list to the defined interface. The command has this syntax:

> **vines access-group** *list*
> **no vines access-group**

The argument *list* is the number of an access list defined with the VINES ACCESS-LIST command.

Example

In this example, the first line permits any two servers (subnet of 1) to talk to each other, regardless of their network numbers, and the second line denies everything else.

```
!
vines routing
vines access-list 1 permit IP 0:1 ffffffff:0    0:1 ffffffff:0
vines access-list 1 deny   IP 0:0 ffffffff:ffff 0:0 ffffffff:ffff
!
```

In this example, the first line prohibits any communication on IPC port number 15 (hex 0F); the second line permits all other communication.

```
vines access-list 1 deny   IPC 0:0 ffffffff:ffff 0xf 0:0 ffffffff:ffff 0xf
vines access-list 1 permit IP 0:0 ffffffff:ffff     0:0 ffffffff:ffff
```

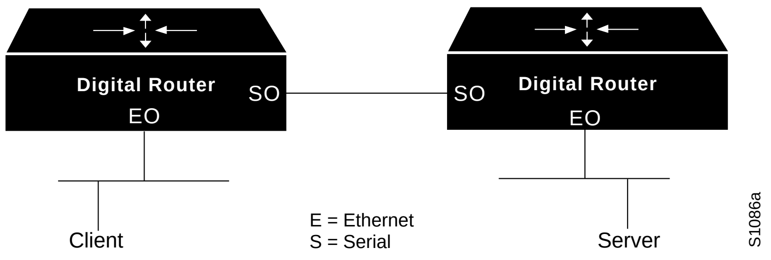
VINES Configuration Examples

Use the following configuration examples to help you configure your VINES routing network.

Propagating Broadcasts

The following examples illustrate how to configure a VINES network where the clients are all behind one router and the servers are behind another router, as depicted in Figure 5–2.

Figure 5–2 VINES Client/Server Configuration



Example for Client

```
!
vines routing
!
interface ethernet 0
vines metric 2
vines serverless
!
interface serial 0
vines metric 15
vines serverless
!
```

Example for Server

```
!
vines routing
!
interface ethernet 0
vines metric 2
!
interface serial 0
vines metric 15
vines serverless
!
```

Filtering Packets

The following example illustrates how to configure an access list that allows packets across two VINES networks and filters all packets across a third VINES network.

Example

```
!
vines routing
vines access-list 1 permit IP 60000A83:0 0:FFFF 60003753:0 0:FFFF
vines access-list 1 deny   IP 0:0 FFFFFFFF:FFFF 0:0 FFFFFFFF:FFFF
!
```

Maintaining the VINES Network

Use these EXEC commands to maintain the VINES routing protocol tables.

Use the privileged EXEC command CLEAR VINES NEIGHBOR to delete the specified address from the neighbor table maintained by the router. This also forces the deletion of all routing table entries that have the given neighbor as the first hop in their paths.

clear vines neighbor {*address* | *****}

The argument *address* is the neighbor address. The asterisk (*****) deletes all entries from the neighbor routing table except the entry for the router itself.

Example

This command clears all entries from the neighbor routing table:

```
gateway# clear vines neighbor *
```

Use the privileged EXEC command CLEAR VINES ROUTE to delete the specified network address from the routing table maintained by the router.

clear vines route {*address* | *****}

The argument *address* is the routing address. The asterisk (*****) deletes all entries from the routing table except the entry for the router itself.

Example

This command clears all entries from the routing table:

```
gateway# clear vines route *
```

Monitoring the VINES Network

Use these EXEC commands to monitor networks configured for VINES routing.

Displaying the VINES Name Table

Use the EXEC command SHOW VINES HOST to display the contents of the VINES name table.

show vines host [*name*]

If the optional *name* argument is included, then only that entry is printed from the table. If no name is present, the entire table is printed.

Displaying VINES Interface Settings

Use the EXEC command SHOW VINES INTERFACE to display all VINES-related interface settings.

show vines interface [*interface unit*]

If no interface name is supplied, the values for all interfaces are printed, along with the VINES global parameters.

Sample output from this command is as follows:

```
 VINES address is 60000A83:1
Next client will be 60000A83:8000
-More-
Ethernet 0 is up, line protocol is up
 VINES protocol processing disabled
-More-
Serial 0 is up, line protocol is up
 Interface metric is 27 (5.4 seconds)
 Outgoing access list is not set
-More-
Serial 1 is up, line protocol is up
 VINES protocol processing disabled
```

Displaying the VINES Neighbor Table

Use the EXEC command SHOW VINES NEIGHBORS to display the entire contents of the VINES neighbor table maintained by this router.

show vines neighbors [*address*]

If the optional *address* argument is supplied, then only that entry from the table is displayed.

Sample output from this command is as follows:

```
Network   Subnet Age   Metric Hardware Address Type      Interface
0027AF92   0001    16       2    0260.8C3C.141C   ARPA      Ethernet0
60000A83   0001    12       2    0000.0C00.0A84   ARPA      Ethernet1
0027AF92   8001    12       2    0000.C05A.0511   ARPA      Ethernet0
60002A2D  0001     -      0    -              -        -
```

Displaying the VINES Routing Table

Use the EXEC command SHOW VINES ROUTE to display the entire contents of the VINES routing table.

show vines route [*address*]

If the optional *address* argument is supplied, then only the routing entry for that network is displayed.

Sample output of this command is as follows:

```
Codes: R - RTP derived, C - connected, S - static, 4 routes
R Net 0027AF92 <LEVEL>(2\{text}) via 0027AF92:1, 21 sec, 0 uses, Ethernet0
R Net 60000A83 [2.H] via 60000A83:1, 503 sec, 0 uses, Ethernet1
C Net 60002A2D is this router's network, 0 uses
```

A `.H` after the metric indicates that the route is currently in a hold-down state.

Displaying VINES Traffic

Use the EXEC command SHOW VINES TRAFFIC to display the statistics kept on VINES protocol traffic.

show vines traffic

Sample output of this command is as follows:

```
   Rcvd: 4218 total, 0 format errors, 0 checksum errors, 12 bad hop count,
         3994 local destination
  Bcast: 3994 received, 2 sent, 1 forwarded
         0 not lan, 0 not gt 4800, 0 no charge
   Sent: 3512 generated, 147 forwarded
         0 encapsulation failed, 65 no route
    IPC: 0 errors received, 12 errors sent, 0 metric sent
   Echo: Received 0, Sent 2
Unknown: 0 packets
```

The VINES PING Command

The EXEC command PING allows the network administrator to determine network connectivity by sending datagrams to the host in question. A sample session follows.

```
Doomgiver# ping coinspinner
Repeat count [5]:
Datagram size [100]:
Timeout in seconds [2]:
Verbose [n]:
Type escape sequence to abort.
Sending 5, 100-byte VINES Echos to 27AF92:1,
timeout is 2 seconds:
!!!!!
Success rate is 100 percent, round-trip min/avg/max = 4/7/8 ms
```

The VINES TRACE Command

The EXEC command TRACE allows the administrator to determine the path that a packet takes when traversing a VINES network. This command does not produce the names of any VINES servers that are traversed. Sample output of tracing through DECbrouter 90 routers to reach a Banyan file server is shown in the following example; the trace test characters are explained in Table 5–3.

```
Doomgiver# trace coinspinner
Type escape sequence to abort.
Tracing the route to COINSPINNER (27AF92:1)
 0 FARSLAYER (60002A2D:1) 0 msec 4 msec 4 msec
 1 COINSPINNER (27AF92:1) 4 msec 4 msec 8 msec
```

Table 5–3 Trace Test Characters

Char	Meaning
nn msec	For each node, the round-trip time for each probe in nn milliseconds.
*	The probe timed out.
?	Unknown packet type.

Debugging the VINES Network

Use these DEBUG commands to obtain reports of VINES' routing functionality. The DEBUG commands are disabled by entering the UNDEBUG version of the command once it is enabled.

debug vines-arp

The EXEC command DEBUG VINES-ARP enables logging of all ARP protocol processing that occurs in the router.

debug vines-echo

The EXEC command DEBUG VINES-ECHO enables logging of all MAC-level echo processing that occurs in the router. This echo functionality is used by the Banyan interface testing programs.

debug vines-packet

The EXEC command DEBUG VINES-PACKET enables logging of general VINES debugging information. This includes packets received, generated, and forwarded, as well as failed access checks and other items.

debug vines-routing

The EXEC command DEBUG VINES-ROUTING enables logging of all RTP update messages sent or received and all routing table activities that occur in the router.

debug vines-table

The EXEC command DEBUG VINES-TABLE enables logging of all modifications to the VINES routing table. This command provides a subset of the information provided by the DEBUG VINES-ROUTING command.

VINES Global Configuration Command Summary

Following is an alphabetical list of the VINES global configuration commands that specify system-wide parameters for VINES support.

[**no**] **vines access-list** *number* {**permit** | **deny**} **IP** *source-address source-mask dest-address dest-mask*

[**no**] **vines access-list** *number* {**permit** | **deny**} *protocol source-address source-mask source-port dest-address dest-mask dest-port*

Creates or deletes a VINES access list. The argument *number* must be an integer between 1 and 100 for both variations of the command. The argument *protocol* is an integer between 1 and 255 or one of these protocols:

- **SPP**—Sequence Packets Protocol
- **RTP**—Routing Table Protocol
- **ARP**—Address Resolution Protocol
- **IPC**—Interprocess Communications
- **ICP**—Internet Control Protocol

Source and destination addresses (*source-address and dest-address*) are VINES addresses written in the standard form.

Source and destination masks (*source-mask and dest-mask*) are also VINES addresses in the standard form, but they indicate the bits in the corresponding address that should be ignored.

Port numbers (*source-port and dest-port*) are integers in the range of 0 to 65535.

The **no** variation with the access list number deletes the access list.

[**no**] **vines host** *name address*

Adds or deletes an entry to the VINES name-to-address mapping table. Once entered, a VINES name can be used anywhere that a VINES addresses is requested.

[**no**] **vines routing** [*address*]

Enables or disables VINES routing. The argument *address* is not necessary if you have Ethernet interfaces, because the router will automatically map itself into the VINES address space. If the optional *address* argument is given, then the routers' VINES address will be forced to be the specified address.

VINES Interface Subcommand Summary

Following is an alphabetical list of the VINES interface subcommands that specify parameters for interfaces configured for VINES routing. These commands must follow an INTERFACE command.

[**no**] **vines access-group** *list*

Assigns or removes an outgoing VINES access list to the defined interface. The argument *list* is the number of an access list defined with the VINES ACCESS-LIST command.

[**no**] **vines arp-enable**

Enables or disables the processing of ARP packets received on the defined interface. When enabled, the router responds to ARP packets and assigns network addresses to clients.

vines encapsulation [**arpa** | **snap**]

Sets the MAC-level encapsulation used for VINES broadcasts on the defined interface. The current defaults are: **arpa** for Ethernet lines, and **snap** for IEEE 802.2 media.

[**no**] **vines metric** [*number*]

Enables or disables the processing of VINES processing on this interface. The system automatically chooses a reasonable metric value, that is used in routing updates and is based upon the delay value set for the defined interface. When the optional *number* argument is supplied, the system forces the VINES delay metric for this interface to the supplied number.

[**no**] **vines redirect-interval** [*number*]

Determines how frequently a router will send an ICP *redirect* message on any given interface. The **no** version restores the default.

[**no**] **vines serverless**

Configures a Banyan VINES network without a server. This command can be used on several routes to build a path to a network that contains servers. The default is NO VINES SERVERLESS.

6 Routing XNS

This chapter describes how to configure your router to perform routing for packets using the Xerox Network System (XNS) protocols.

You will find information about the following topics and tasks:

- How to configure a routing process for XNS routing. This includes information on the principal XNS protocols, XNS addressing, configuring your router to route XNS traffic, managing security issues, and maximizing performance.
- How to configure helper addresses and flooding options for broadcast traffic.
- How to configure options for access lists and filters.
- Configuration restrictions and requirements for encapsulation of all important lower- layer protocols, including Ethernet, serial ports and others.

This chapter also contains configuration information on Ungermann-Bass' and 3Com's XNS protocols.

The section XNS Configuration Examples includes examples of actual configurations for working XNS networks, including Ungermann-Bass and 3Com.

Additionally, EXEC-level commands for monitoring and debugging the XNS network are described. These commands are found in the last sections of the chapter, along with concise summaries of the global and interface-specific configuration commands.

The DECbrouter 90 Implementation of XNS

The DECbrouter 90 provides a subset of the XNS protocol stack to support XNS routing on its routers. The same DECbrouter 90 that routes XNS can also route another protocol stack like TCP/IP or DECnet. At the Physical and Data Link Layers, XNS traffic can be routed over Ethernets, or point-to-point serial lines running HDLC or LAPB, or X.25, Frame Relay, or SMDS networks.

Ethernet

When XNS routing is enabled, the address used is either the IEEE-compliant address specified in the XNS routing configuration command or the first IEEE-compliant address in the system. The address also is used as the node address for non-LAN media, notably serial links.

XNS Addresses

Both Data Link (MAC) and Network Layer addressing are needed in any network that supports routing; XNS is no exception. An XNS Network Layer address is composed of three fields:

- The network number uniquely identifies a network in an internet.
- The host number uniquely identifies a host on a network.
- The socket number uniquely identifies a socket within the operating system of the host. A socket is a transport address that is the source or destination of packets.

Network and Host Numbers

The network number is expressed in decimal format in DECbrouter 90 configuration files and routing tables. When configuring a DECbrouter 90, enter the network number in decimal.

Addresses must be unique throughout an XNS internet. Because both the network number and the host address are needed to deliver traffic to a host, addresses are usually given as network numbers, followed by host addresses, separated with dots. An example address follows.

Example

In this example, the network number is 47 (decimal), and the host address is 0000.0c00.23fe (hex). A network number of 0 is used when the network number is unknown; this is the local network number.

```
47.0000.0c00.23fe
```

Socket Numbers

An XNS socket number is a 16-bit field in the IDP header. Sockets are selected by the client processes of each host before a connection is established. Certain socket numbers are considered to be well-known sockets (WKS), meaning that the service performed by the software using them is statically defined. Each system element supplying a specific well-known service does so at the same WKS. Socket numbers above the well-known range can be selected and reused at random.

Configuring XNS

Follow these steps to configure XNS routing:

1. Enable XNS routing using the XNS ROUTING command.
2. Assign a unique XNS network number to each interface using the XNS NETWORK command.
3. Optionally configure performance parameters and helper addresses (which help manage broadcast traffic).
4. Optionally configure access lists and filters.

There are specific configurations for the Ungermann-Bass Net/One networks; these are described in a later section.

Enabling XNS Routing

The first step in the configuration process is to specify XNS as the protocol you are enabling. Use the XNS ROUTING global configuration command to enable XNS routing. The full syntax of this

xns routing [*address*]
no xns routing

The optional argument *address* is used as the router's address on interfaces that have no native MAC-level addresses (principally serial lines). The NO XNS ROUTING command disables all XNS processing.

If the argument *address* is omitted, the router uses the address of the first IEEE-compliant (Ethernet) interface hardware address it finds in its interface list. The address 0123.4567.abcd is used as a default for non-IEEE interfaces.

Next, use the XNS NETWORK interface subcommand to assign a decimal XNS network number to each interface and enable each interface to run XNS protocols. The full syntax of the command follows.

xns network *number*
no xns network

The argument *number* is the network number in decimal format. Interfaces not enabled to run XNS ignore any XNS packets that they receive. Every XNS interface in a system must have a unique XNS network number.

The **no** version of the command deassigns the XNS network number for the interface and disables XNS on that interface.

Example

This example starts XNS routing and assigns XNS network numbers to the physical networks connected to two of the router's serial interfaces.

```
xns routing
interface serial 0
xns network 20
interface serial 1
xns net 21
```

Configuring Static Routing

To add a static route from your router to a remote destination in the XNS routing table, use the XNS ROUTE global configuration command. The full syntax follows.

xns route *network host-address*
no xns route *network host-address*

The argument *network* is the destination XNS network number in decimal. The argument *host-address* is a decimal XNS network number and a hexadecimal host number, separated by a dot. The argument *host-address* must refer to a node directly connected to one of the router's physical networks.

Static routes usually are not used in XNS environments, because nearly all XNS routers support dynamic routing with RIP. Dynamic routing is enabled by default in the DECbrouter 90. The **no** version of the command removes the static route.

Example

The following example sets up a host (router) address of 21.0456.acd3.1243 as the static recipient of packets destined for network 25:

```
xns route 25 21.0456.acd3.1243
```

Managing Throughput

You have several options for managing throughput while dynamically routing. You can set up multiple paths and send packets over these paths in a round-robin fashion. You also can enable or disable fast switching. You even can adjust how often your router uses RIP to send routing table updates to its neighbors, provided all the neighbors are DECbrouter 90 routers.

Setting Multiple Paths

A DECbrouter 90 router can use multiple paths to reach an XNS destination in order to increase throughput in the network. By default, the router will pick one best path and send all traffic on this path, but you can configure it to remember two or more paths that have equal costs (as computed by the routing protocol; hop count for standard XNS RIP) and balance the traffic load across all the available paths.

To set the maximum number of multiple paths used for any single destination, use the XNS MAXIMUM-PATHS global configuration command. The full syntax follows.

xns maximum-paths *paths*
no xns maximum-paths

The argument *paths* is the number of paths to be assigned. The default value for *paths* is 1. Packets are distributed over the multiple paths in round-robin fashion on a packet-by-packet basis. To disable multiple paths, use the command with the default value of 1. The NO XNS MAXIMUM-PATHS command restores the default.

The EXEC command SHOW XNS ROUTE displays the entire routing table, so you can examine your additional routes and the maximum path cost for each. See the section Monitoring an XNS Network when you are ready to use this command.

Example

The following example asks the router to send packets for each destination over two alternate paths, if available:

```
xns maximum-paths 2
```

Configuring XNS Fast Switching

XNS fast switching achieves higher throughput by using a cache created by previous transit packets. Fast switching for XNS provides load sharing on a per-packet basis. When you enable XNS routing, fast switching is automatically enabled as well. Use the NO XNS ROUTE-CACHE interface command to disable fast switching. The full syntax of the command follows.

xns route-cache
no xns route-cache

Use the NO XNS ROUTE-CACHE command to disable fast switching and the xns route-cache interface subcommand to re-enable fast switching.

Adjusting Timers

XNS RIP sends routing table updates to its neighbor routers every 30 seconds, unless you specify some other time interval. You can reset the routing update timers on a per-interface basis using the XNS UPDATE-TIME interface subcommand:

xns update-time *seconds*

Note

This command does not affect the Ungermann-Bass routing protocol.

XNS routing timers are affected by the value set for the *seconds* argument:

- Routing updates are sent through the affected interface.
- XNS routes are marked invalid and placed in holddown if no routing updates for those routes are heard within three times the value of the update timer.
- XNS routes are removed from the routing table if no routing updates are heard within six times the value of the update timer.

There is not a **no** version of this command. If you want to return to the default condition, use the XNS UPDATE-TIME command with a value of 30 for the *seconds* argument. The minimum is ten seconds.

Note

Be careful with this command. Use it only in an environment with all DECbrouter 90 or Cisco routers. . Make sure that all timers are the same for all routers attached to the same network segment.

The EXEC command SHOW XNS ROUTE displays the value of these timers. See the section Monitoring an XNS Network chapter when you are ready to use this command.

Configuring Ungermann-Bass Net/One XNS

This section describes the protocols and configurations specific to Ungermann-Bass' Net/One networks. Net/One end nodes communicate using the XNS protocol, but there are a number of differences between Net/One's usage of the protocol and the usage common among other XNS nodes. This section discusses the Net/One implementation and how to configure routers to work in Net/One networks.

Ungermann-Bass Net/One on a Digital Router

Ungermann-Bass's Net/One products use XNS at the network layer. However, Net/One as a whole is not equivalent to standard XNS. When you use DECbrouter 90 in Net/One environments, you must consider a few factors. The following are the important differences between the Net/One environment and more common XNS usage:

- Net/One routers use an Ungermann-Bass proprietary routing protocol instead of standard XNS RIP. Although they generate both Ungermann-Bass update packets and standard RIP update packets, Net/One routers only listen to Ungermann-Bass updates. The DECbrouter 90 supports both reception and generation of Ungermann-Bass routing packets, and the DECbrouter 90 can interoperate with Ungermann-Bass routers.
- Net/One routers send periodic Hello packets, which are used by end nodes in discovering routers to be used in sending packets to destinations that are not on the local cable. Ordinary XNS end hosts use RIP for this purpose. The DECbrouter 90 can be configured to generate Ungermann-Bass Hello packets.
- Net/One equipment uses a non-XNS protocol for network software downloads. During the downloading process, XNS network numbers are embedded in the packets of this protocol. Ungermann-Bass routers pass the booting protocol from network to network and modify the embedded network numbers. The DECbrouter 90 does not understand the Net/One booting protocol and does not modify the embedded network numbers. For network booting to work through the DECbrouter 90, Net/One Network Management Consoles must be specially configured. Contact Ungermann-Bass to find out how to configure your Ungermann-Bass Network Management Console.
- The Ungermann-Bass Net/One Network Resource Monitor (a network management and monitoring tool) uses XNS packets whose destination host addresses are specific nodes, but whose destination network addresses are broadcast (network -1). These packets are sent as MAC-layer broadcasts, and are expected to be flooded throughout the XNS internet. Flooding of these packets can be enabled on the DECbrouter 90 with the XNS FLOOD SPECIFIC ALLNETS interface configuration subcommand, which is described in the section Handling XNS Broadcasts, Flooding, and Helpering.
- Net/One equipment uses a set of proprietary network management protocols. The DECbrouter 90 does not participate in these protocols.

Configuring Ungermann-Bass Net/One Routing

To enable Ungermann-Bass Net/One routing, use the XNS UB-EMULATION global configuration command. The full syntax of this command follows.

xns ub-emulation
no xns ub-emulation

This command causes Hello packets and routing updates in Ungermann-Bass format to be sent out through all the interfaces on which XNS is enabled. It also causes the DECbrouter 90 to get its routing information for remote networks from Ungermann-Bass updates. Unless the XNS HEAR-RIP command is issued, the DECbrouter 90 will completely ignore standard RIP updates and will behave like an Ungermann-Bass router.

The **no** version of the command restores the router to standard XNS routing mode.

If you need your DECbrouter 90 to receive RIP updates on some interfaces, you can configure it to do so using the XNS HEAR-RIP interface configuration subcommand:

xns hear-rip [*access-list*]
no xns hear-rip

The XNS HEAR-RIP command causes the DECbrouter 90 to treat standard RIP updates incoming on the interface as if they were Ungermann-Bass updates, with delay metrics computed as though each hop mentioned by the RIP update were a 9.6-Kbps serial link. Ordinarily, the result is that the DECbrouter 90 will prefer an all-Ungermann-Bass path to an all-RIP path. If you only want certain routes to be learned through standard RIP, you can specify an XNS access list as an argument to the XNS HEAR-RIP command. The DECbrouter 90 will learn from RIP packets only routes to networks permitted by the access list. Note that the access list is applied to individual routes within the RIP packet, not to the address of the packet's sender. The **no** version of the command causes the interface not to receive RIP updates.

In an Ungermann-Bass environment, all interfaces should be configured with the XNS FLOOD BROADCAST ALLNETS and XNS FLOOD SPECIFIC ALLNETS interface configuration subcommands. Interfaces should *not* be configured with XNS FLOOD BROADCAST NET-ZERO.

The commands XNS FLOOD BROADCAST ALLNETS and XNS FLOOD SPECIFIC ALLNETS are described in the section Handling XNS Broadcasts, Flooding, and Helpering.

An Ungermann-Bass configuration example is found in the section XNS Configuration Examples.

Ungermann-Bass Routing Protocol and Interactions with RIP

The routing protocol used by Net/One routers is a distance-vector or Bellman-Ford protocol, similar to standard XNS RIP. The major difference between the two protocols lies in the metrics used. While standard RIP uses a hop count to determine the best route to a distant network, the Ungermann-Bass protocol uses a path-delay metric. The standard RIP protocol maintains information only about hop counts, while the Ungermann-Bass protocol maintains information both about hop counts and about its own metrics.

Ungermann-Bass routers generate standard RIP updates by extracting the hop-count values from the Ungermann-Bass routing protocol. When configured in Ungermann-Bass emulation mode, the DECbrouter 90 participates in this protocol, and behave (insofar as routing protocols are concerned) like Ungermann-Bass routers.

The DECbrouter 90 can also be configured to listen to standard RIP updates when in Ungermann-Bass emulation mode, using the XNS HEAR-RIP interface configuration subcommand. When a the DECbrouter 90 in Ungermann-Bass emulation mode receives a RIP packet, each route in that packet is treated as though it had come from an Ungermann-Bass routing packet. The hop count used is the actual hop count from the RIP packet. The delay metric used is computed by assuming that each hop is the longest-delay link used by Ungermann-Bass: a 9.6-kbps serial link. Information from RIP packets is used in creating outgoing Ungermann-Bass updates, and vice versa.

Handling XNS Broadcasts, Flooding, and Helpering

This section explains how XNS handles broadcasts, and how flooding and helpering help you manage broadcast traffic.

Overview of XNS Broadcasts

Network end nodes often use broadcast packets for service discovery; a request is broadcast to many or all nodes in the internet, and one or more of the nodes that can offer the needed service reply to the broadcast. Both end nodes and routers sometimes use broadcast packets to contain data that must be received by many other nodes; an example would be a RIP routing update.

Although broadcasts can be very useful, they are not without costs. Every node on a physical network must receive and process all broadcasts sent on that network, even if the processing consists of ignoring the broadcasts. If many nodes answer the broadcast, network load might increase dramatically for a short period of time. If the broadcast is propagated to more than one physical network, extra load is presented on all the networks involved and on all the intervening routers.

The following describes types of broadcasts and how each is handled:

- A *local* broadcast is one that is intended only for nodes on the physical network (typically one Ethernet LAN) on which the packet is originally sent. XNS networks usually denote local broadcasts by a specific network number in the packet's destination field. If a node does not know the number of the local XNS network (common at bootup time), it can use a network number of zero to denote a local broadcast.
- An *all-nets* broadcast is one that is intended for all nodes throughout the XNS internet. XNS networks usually denote all-nets broadcasts by an all-ones destination network field (typically written as -1 or as FFFFFFFF).
- A *directed* broadcast is one that is intended for all nodes on a specific physical network other than the network on which the packet originates. Directed broadcasts are denoted by the use of a specific network number, other than that of the network on which the packet originates, in the destination field.

All these broadcast types use the node address FFFF.FFFF.FFFF in the packet's destination node field. The destination MAC address used in the underlying LAN frame is the broadcast address. Directed broadcasts intended for remote networks can be sent directly to the MAC address of a router that provides the path to their ultimate destination, and physically broadcast only when they reach it.

Flooding and Helpering

Some implementations expect all broadcasts to be treated as local broadcasts. Others expect broadcasts with zero destination network fields to be treated as all-nets broadcasts. Some do not support directed broadcasts. In addition, some implementations expect packets with all-ones destination network fields, but with destination node fields that correspond to specific hosts, to be flooded throughout the internet as MAC-layer broadcasts. This way, nodes can be located without knowledge of which physical networks they are connected to. Digital supports all these models by using *helpering* and *flooding* features.

Helpering, which is typically used for service discovery broadcasts, sends the broadcasts to user-specified candidate servers on remote networks. When a packet is helpered, the router changes its destination address to be the configured helper address, and it is routed toward that address. The host at the helper address is expected to process the packet and (usually) to reply to the packet's sender. A helper address can be a directed broadcast address, in which case the helpered packet will be forwarded to a remote network and rebroadcast there.

Flooding sends packets throughout the entire XNS internet. Flooded packets are not modified, except for hop count bookkeeping fields. Flooding is useful in cases where many nodes throughout the internet need to receive a packet or where a service that can be anywhere in the internet must be discovered. Avoid flooding in large, slow, or heavily loaded networks, because the load presented to routers, links, and end nodes by heavy flooded traffic is large.

Configure XNS helpering using the XNS HELPER-ADDRESS interface configuration command and the XNS FORWARD-PROTOCOL global configuration command.

xns helper-address *host-address*
no xns helper-address *host-address*

The argument *host-address* is a dotted combination of the network and host addresses as explained in the XNS ROUTE command. Broadcasts received on this interface are considered candidates for helpering as described previously. The **no** version of the command disables XNS helpering.

The XNS FORWARD-PROTOCOL global configuration command allows you to specify which XNS protocols will be considered for helpering when received in broadcast packets. The full syntax of the command is as follows:

xns forward-protocol *type*
no xns forward-protocol *type*

The argument *type* is a decimal number corresponding to an appropriate XNS protocol. Only packets with types listed in XNS FORWARD-PROTOCOL commands are considered candidates for helpering. See the documentation accompanying your host XNS implementation to determine the protocol type number.

Use the NO XNS FORWARD-PROTOCOL command and the appropriate argument to disable the forwarding of the specified protocol.

Whenever a DECbrouter 90 receives an XNS broadcast packet, it processes the packet as follows:

- If the packet is a routing update or requests services that are offered by the DECbrouter 90 itself, the packet is processed by the router and is not forwarded any further.
- If a helper address is set on the interface on which the packet arrived, and the packet's protocol type appears in the **xns forward-protocol** list, the packet is forwarded to the helper address. The helper address can be a directed broadcast address.
- If the packet is neither locally serviceable nor a candidate for helpering, it is considered for flooding. Three classes of packets can be flooded:
  —Packets with destinations of -1.FFFF.FFFF.FFFF (enabled with XNS FLOOD BROADCAST ALLNETS)
  —Packets with destinations of 0.FFFF.FFFF.FFFF (enabled with XNS FLOOD BROADCAST NET-ZERO)
  —Packets with destinations of -1.*specific-host* (enabled with XNS FLOOD SPECIFIC ALLNETS)

The XNS flooding interface configuration commands are as follows:

xns flood broadcast allnets
no xns flood broadcast allnets

xns flood broadcast net-zero
no xns flood broadcast net-zero

xns flood specific allnets
no xns flood specific allnets

All of the preceding are interface configuration subcommands and apply to flooding of packets *received* on the interfaces for which they are specified.

Different XNS host implementations require different flooding and helpering behavior. By default, the DECbrouter 90 do no flooding at all. It is most closely in accordance with the XNS specification to set XNS FLOOD BROADCAST ALLNETS and XNS FLOOD SPECIFIC ALLNETS, but not to set XNS FLOOD BROADCAST NET-ZERO. 3Com environments often require flooding of net-zero broadcasts.

Digital chooses the interfaces through which flooded packets are sent according to rules designed to avoid packet looping and most packet duplication. The underlying principle of these rules is that packets should be flooded *away* from their sources, never *toward* them. Packets that the DECbrouter 90 is configured to flood are sent out through all interfaces, except in the following cases:

- Packets that would ordinarily be flooded are ignored unless they are received via the interface that would be used to route a unicast packet to the flooded packet's *source* network. If there are multiple paths to the source network, only packets received on the primary path (the first path the DECbrouter 90 learned) are flooded. If a packet is received on an interface that fails this rule, the interface that passes it will receive another copy of that packet.

- Packets are never flooded *out* of the DECbrouter 90 through any interface that is one of the router's paths back toward the packet's source. A copy of the flooded packet will appear on the network connected to such an interface via some other path.
- If the DECbrouter 90 has no route to a packet's source network, the packet is not flooded. This is to prevent odd behavior during routing convergence after network topology changes.
- Packets that fail the access lists applied to outgoing interfaces are not flooded through those interfaces. Users can use access lists to get control over flooding behavior.

Packets with -1 destination networks and specific destination hosts are sent as MAC-layer broadcasts so that they can be picked up and further flooded by other routers.

For backward compatibility, any attempt to set a helper address of -1.FFFF.FFFF.FFFF on an interface will result in that interface having no helper address set, but having XNS FLOOD BROADCAST ALLNETS enabled.

Both helpering and flooding increment packets' hop-count fields.

Notes and Tips on Configuring XNS Helpering

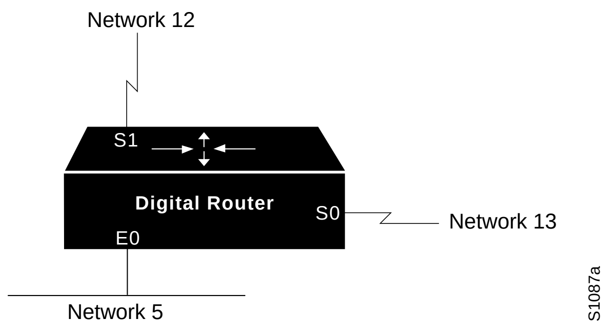
You need to decide how you want the router to handle different kinds of broadcast packets. This is an excellent opportunity for you to reduce unnecessary network traffic, if you think carefully about all your options.

Many broadcasts occur when a node first becomes active on the network. A host will generate a broadcast packet when it does not know the current address of whatever host is supposed to receive its next packet—the local server, for instance. Generally, it is not a good idea to place a router between users and the servers that carry their primary applications; you should minimize internet traffic. However, if you need that server configuration for some other reason, you need to ensure that users can broadcast between networks without cluttering the internet with unnecessary traffic.

You can configure your routers to block all broadcasts. You have more than simply this all-or-nothing choice, however, by using the XNS HELPER-ADDRESS, XNS FORWARD-PROTOCOL, and XNS FLOOD commands.

Ignore the protocol-filtering issue for a moment and just consider some examples of how helper addresses are used. In Figure 6–1, the E0 interface has a helper address set, with the helper on network 12 available through the S1 interface. Some broadcast packets are being received on this E0 interface.

Figure 6–1 Helper Addresses



- A broadcast packet with a network address of 5 will be forwarded to the helper address on network 12.
- A broadcast packet addressed to network 0 will also be forwarded to the helper address on network 12.
- A broadcast packet addressed to network 13 is a directed broadcast, and will be sent through the E1 interface directly to network 13. It will not be sent to the helper address.

Forwarding Specific Protocols

In Figure 6–1, the E0 interface is set to a helper address and forwarding for protocol 1 only. (The protocol numbers will vary depending on your XNS host implementation.) Broadcast packets are *arriving* on the E0 interface from network 5.

- A broadcast packet destined for network 5 and protocol 1 will be sent to the helper address.
- A broadcast packet destined for network 5 and another protocol is discarded because it fails the protocol test.
- A broadcast packet destined for network 0 and protocol 1 is sent to the helper address.
- A broadcast packet destined for network 0 and a protocol other than 1 is discarded.

A broadcast packet destined to network 12 is a directed broadcast, and is sent out, still in broadcast form, on the S1 interface to network 12. This has nothing to do with the helper-address or protocol forwarding.

Example

In this example, a helper address is specified and protocol type 2 is forwarded to that address.

```
xns forward-protocol 2
interface ethernet 0
xns helper-address  26.FFFF.FFFF.FFFF
```

Configuring XNS Access Lists and Filters

You can configure XNS access lists to filter traffic on XNS interfaces.

An access list is defined by a series of commands. Each access list entry contains only one address parameter, and the list must be within the range of 400 to 499. How this address is interpreted is defined by the command that will use the list.

XNS access lists filter on the source and destination addresses only. This means that they can prevent traffic from going to or coming from specific hosts. Extended XNS access lists are numbered from 500 to 599 and filter on XNS protocol and socket fields. The extended filters can prevent entire classes of packets (SPP, Echo, and so on) from passing a router interface and can also filter traffic going to or coming from specific processes.

As with all other DECbrouter 90 access lists, an implicit *deny everything* is defined at the end of the list. If this is not desired, an explicit *permit everything* definition must be included at the end of the list. Complex configuration examples are shown in the section XNS Configuration Examples.

Configuring XNS Access Lists

To configure an XNS access list, use the ACCESS-LIST global configuration command with the following syntax:

> **access-list** *number* {**deny** | **permit**} *XNS-source-network*. [*source-address* [*source-mask*]] *XNS-destination-network*. [*destination-address*[*destination-mask*]]
> **no access-list** *number*

Note

For typographical reasons, access list command syntax in this section is shown on multiple lines; an access list command must be on a single line when given as a configuration command.

The argument *number* must be a number between 400 and 499. The **no** version of the command removes the entire access list.

The only required parameter for standard XNS access lists is the XNS source network address. The rest of the parameters are optional except that the source and/or destination address masks are present only if the corresponding source and/or destination address was entered. Note that XNS uses MAC addresses as the node ID; see the examples for further clarification.

Example 1

This example denies access from source network -1 (all XNS networks) to destination network 2. All other packets are permitted.

```
access-list 400 deny -1 2
access-list 400 permit -1
```

Example 2

This example denies access from XNS source address 21.0000.0c00.1111. The destination network does not make any difference. All other packets are permitted.

```
   access-list 400 deny 21.0000.0c00.1111
   access-list 400 permit -1
```

Example 3

This example denies access from all hosts on network 1 that have a source address beginning with 0000.0c. All other packets are permitted.

```
   access-list 400 deny 1.0000.0c00.0000 0000.00ff.ffff
   access-list 400 permit -1
```

Example 4

In the following example, we deny access from source address 0011.1622.0015 on network 21 to destination address 01D3.020C.0022 on network 31. All other packets are permitted.

```
   access-list 500 deny 1 21.0011.1622.0015 0000.0000.0000 31.01D3.020C.0022
      0000.0000.0000
```

Configuring Extended Access Lists

For extended access lists, the ACCESS-LIST command again must be typed on one line using this syntax:

> **access-list** *number* {**deny** | **permit**} *xns-protocol source-network.* [*source-address* [*source-mask*]] *source-socket destination-network.* [*destination-address* [*destination-mask*]] *destination-socket*
> **no access-list** *number*

The argument *number* must be a number between 500 and 599. The **no** version of the command removes the entire access list.

The source and destination addresses and masks are optional. The protocol number *xns-protocol* is the only required parameter. A network number of -1 matches all networks; a socket number of 0 matches all sockets. The **no** version of the command removes the entire access list.

Example 1

This example denies access to packets with protocol 5 from source network 1, source socket 1234 that are trying to be routed to destination network 2, destination socket 1234:

```
   access-list 500 deny 5 1 1234 2 1234
```

Example 2

This example adds masks for the source and destination networks:

```
access-list 500
permit 1 21.0011.1622.0015 0000.0000.0000 1234 31.01D3.020C.0022
   0000.00ff.ffff 1234
```

Filtering Outgoing Packets

An XNS access list group number is assigned to an interface with the XNS ACCESS-GROUP interface subcommand. The full syntax of this command follows.

xns access-group *number*
no xns access-group *number*

The argument *number* specifies the access list number defined by the XNS global ACCESS LIST command. This command causes all outgoing XNS packets that would ordinarily have been forwarded by the router through this interface to be compared to the access list. If the packet fails the access list, it is not transmitted, but is discarded instead.

Example

This example uses the XNS ACCESS-LIST command to deny access to protocol 1 from source network 1, source socket 1234 to destination network 2, destination socket 1234, then assigns number 500 to a group to be filtered:

```
access-list 500 deny 1 1 1234 2 1234
access-list 500 permit -1
!
interface ethernet 0
xns access-group 500
```

Filtering XNS Routing Updates

This section describes the filtering commands that use access lists to control which routing information is accepted or passed on within XNS networks. The commands filter incoming traffic, outgoing routing information, and specific routers.

Input Filters: Adding to the Routing Table

To control which networks are added to your router's routing table, use the XNS INPUT-NETWORK-FILTER interface subcommand.

xns input-network-filter *access-list-number*
no xns input-network-filter *access-list-number*

The argument *access-list-number* is the access list number specified in the XNS ACCESS-LIST command.

Example

In this example, access list 476 controls which networks are added to the routing table when RIP packets are received. The address in the access list is the address of the network that you want to be able to receive routing updates about.

```
access-list 476 permit 16
interface ethernet 0
xns input-network-filter 476
```

This set of configuration commands ensures that network 16 is the only network whose information will be added to the routing table from Ethernet 9.

Output Filters: Controlling the List of Networks

To control the list of networks that your router sends out in routing updates, use this interface subcommand:

xns output-network-filter *access-list-number*
no xns output-network-filter *access-list-number*

The argument ACCESS-LIST-NUMBER is the access list number specified in the XNS access-list command.

Example

In the following example, access list 496 controls which networks are sent out in routing updates. The second line specifies interface serial 1, and the third line causes network 27 to be the only network advertised in routing update packets. Information about other networks will not be advertised in routing updates (for the specified interface only; other interfaces can advertise the full routing table).

```
access-list 496 permit 27
interface serial 1
xns output-network-filter 496
```

Router Filters

To control the list of routers from which data will be accepted, use the XNS ROUTER-FILTER interface subcommand:

xns router-filter *access-list-number*
no xns router-filter *access-list-number*

The argument *access-list-number* is the access list number specified in the XNS ACCESS-LIST command.

Example

In this example, access list 466 defines the only router from which data will be accepted. In this case, the address parameter is the address of a router.

```
access-list 466 permit 26.0000.000c0.047d
interface serial 0
xns router-filter 466
```

Information from a disallowed router is ignored.

XNS Configuration Examples

This section includes examples of common configurations, and is designed to help you put all the specific command information into a complex, real-world configuration file. Included are basic configuration examples, examples of flooding, and both simple and complex access lists. Some of the examples refer to 3Com or Ungermann-Bass XNS, rather than standard XNS; all examples are clearly marked.

Creating a Routing Process

The following example establishes XNS routing on a DECbrouter 90 (creating a routing process), then names three interfaces and gives them their individual network numbers. The router will use the preassigned MAC-level addresses as XNS node addresses on the Ethernet interface. The serial interfaces will use the MAC address associated with the first IEEE 802.x interface found on the router.

Example

```
xns routing
!
interface ethernet 0
xns network 20
!
interface serial 0
xns network 21
!
interface serial 1
xns network 24
```

Setting Timers

This example creates a routing process that specifies a specific address for use on serial lines and other non-802.x interfaces. After the address is specified, you can specify the interfaces and assign them network numbers. The RIP routing update timers for the serial and the Ethernet interfaces also have been changed.

Example

```
xns routing 0000.0C53.4679
!
interface ethernet 0
xns network 20
xns update-time 20
!
interface serial 0
xns network 24
xns update-time 40
!
interface serial 1
xns network 21
xns update-time 25
```

Configuring for Multiprotocol Routing

This example illustrates one way of enabling XNS and another protocol. Do a pencil copy of your proposed configuration commands and check them against the other protocol chapters in this manual before you proceed with configuring more than one protocol in a session.

Example

```
xns routing
novell routing
!
interface ethernet 0
xns network 20
novell network 4e
!
interface serial 1
xns network 21
novell network 6bb
!
interface serial 0
xns network 24
novell network 4ad
```

Configuring for Ungermann-Bass Routing

In this example, basic Ungermann-Bass Net/One routing is enabled by the following steps:

1. Enabling XNS routing
2. Specifying Ungermann-Bass routing
3. Defining the interfaces and networks

Interface serial 0 is connected to a non-Ungermann-Bass part of the XNS internet, so the XNS HEAR-RIP command is used to allow the learning of routes from the standard RIP updates used by the remote routers. There are Ungermann-Bass nodes connected to interface Ethernet 0. Broadcast flooding is configured to match the expectations of Ungermann-Bass software.

Example

```
xns routing
xns ub-emulation
!
interface Ethernet 0
xns network 23
xns flood broadcast allnets
xns flood specific allnets
!
interface serial 1
xns network 21
xns flood broadcast allnets
xns flood specific allnets
!
interface serial 0
xns network 24
xns hear-rip
xns flood broadcast allnets
xns flood specific allnets
```

3Com Access List

This partial example permits and denies specific services between networks 1002 and 1006 in a 3Com network. Echo and error packets can go from 1002 to 1006, as well as all SPP and PEP (normal data traffic). However, all NetBIOS requests are denied. The final three lines are blanket permissions for RIP, SPP, and PEP, because access lists assume you wish to deny anything you do not specifically permit. These blanket permissions must come at the end of the list, as shown in the example configuration.

Example

```
access-list 524 permit 2 1002 0x0000 1006 0x0000
!  permit Echo from 1002 to 1006
access-list 524 permit 3 1002 0x0000 1006 0x0000
!  permit Error from 1002 to 1006
access-list 524 deny 5 -1 0x0000 -1 0x046B
!  deny all NetBIOS
access-list 524 permit 4 1002 0x0000 1006 0x0000
!  permit PEP from 1002 to 1006
access-list 524 permit 5 1002 0x0000 1006 0x0000
!  permit SPP from 1002 to 1006
access-list 524 permit 1
!  permit all RIP
!
!These are needed if you want PEP and SPP to be permitted from
!networks other than 1002
access-list 524 permit 4
!  permit all PEP
access-list 524 permit 5
!  permit all SPP
```

Monitoring an XNS Network

Use the EXEC commands described in this section to obtain displays of activity on your XNS network.

Displaying Cache Entries

Use the SHOW XNS CACHE command to display a list of fast-switching cache entries. Enter this command at the EXEC prompt:

show xns cache

Displaying Interface Parameters

Use the SHOW XNS INTERFACE command to display interface-specific XNS parameters. Enter this command at the EXEC prompt:

show xns interface [*name*]

The optional argument *name* can be used to request a display of a particular interface.

The following sample output shows a display of all interface statistics.

```
Ethernet 0 is up, line protocol is up
XNS address is 60.0000.0c00.1d23
xns encapsulation is ARPA
Helper address is 912.ffff.ffff.ffff
Outgoing address list is not set
Input filter list is not set
Output filter list is not set
Router filter list is not set
Update timer is not set
XNS fast-switching enabled

Serial 0 is administratively down, line protocol is down
XNS protocol processing disabled

Serial 1 is up, line protocol is up
XNS protocol processing disabled
```

In this display, the Ethernet 0 interface has a helper address set, but all other parameters are set to the defaults. The Update Timer refers to itself as not set when it is set to default value. The serial 0 interface is down, and XNS processing is disabled. The serial 1 interface is up, but it is not processing XNS packets.

For the serial 0 and serial 1 interfaces, the NO XNS NETWORK command is in force. You can enable XNS processing by using the XNS NETWORK configuration command.

Displaying the Routing Table

Use the EXEC command SHOW XNS ROUTE to display the entire XNS routing table. Enter this command at the EXEC prompt:

show xns route *network-number*

The optional network number argument specifies a particular network.

Following is sample output:

```
Codes: R - RIP derived, C - connected, S - static, 1 learned routes

Maximum allowed path(s) are/is 1
C Net 14 is directly connected, 0 uses, Ethernet0
C Net 15 is directly connected, 0 uses, Ethernet1
R Net 16 [1/0] via 14.0000.0c00.3e3b,  10 sec, 0 uses, Ethernet0
```

In this display, RIP-derived routes are indicated by the letter R. Networks that are reachable are listed in numerical order within each category—RIP or connected, in this case.

The routing table also tells you the address of the router that constitutes the first hop in the route (always the same router in this case), the round-trip delay encountered on the route, and the interface through which the route is available.

Displaying Traffic Statistics

Use the EXEC command SHOW XNS TRAFFIC to display packet statistics, including packets sent, received, and forwarded. Enter this command at the EXEC prompt:

show xns traffic

Sample output follows. Table 6–1 describes the fields displayed.

```
Rec: 3968 total, 0 format errors, 0 checksum errors, 0 bad hop count,
3968 local destination,  0 multicast
Bcast: 2912 received, 925 sent
Sent:  5923 generated, 500 forwarded, 0 encapsulation failed, 0 not routable
Errors:  10 received, 20 sent
Echo:  Recd:  100 requests, 89 replies   Sent:  20 requests, 20 replies
Unknown:  5 packets
```

Table 6–1 XNS Traffic Statistics Field Descriptions

Field	Description
Rec:	The total number of XNS packets received by the router.
format errors	Number of packets received and discarded because of format errors in the header.
checksum errors	Number of packets received and discarded because of checksum errors.
bad hop count	Number of packets discarded because their hop count fields were equal to or greater than 16.
local destination	Number of packets received on the interface that were broadcast or were destined for internal processing by the router, including helpering or flooding.
multicast	Number of packets received with multicast addresses.
Bcast:	Number of broadcast packets received and sent.
Sent:	
generated	Number of packets created by the router itself.
forwarded	Number of packets received from other nodes and forwarded.
encapsulation failed	Number of packets discarded because encapsulation could not be accomplished.
not routable	Number of packets discarded because the router had no path to their destination networks.
Errors:	Number of Error packets sent and received.
Echo:	Number of Echo packets received and sent, specifying how many replies were received.
Unknown:	Packets destined for local processing, but which the router could neither handle, helper, nor flood.

Debugging an XNS Network

Use the EXEC commands described in this section to troubleshoot and monitor your XNS network. For each DEBUG command listed, there is a corresponding UNDEBUG command that turns off the message logging.

The command DEBUG XNS-PACKET enables logging of XNS packet traffic, including the addresses for source, destination, and next hop router of each packet.

```
debug xns-packet
```

The command DEBUG XNS-ROUTING displays XNS routing transactions.

```
debug xns-routing
```

XNS Global Configuration Command Summary

Following is an alphabetically arranged list of the XNS global configuration commands.

[**no**] **access-list** *number* {**deny** | **permit**} *XNS-protocol source-network.* [*source-address* [*source-mask*]] *source-socket destination-network.* [*destination-address* [*destination-mask*]] *destination-socket*

no access-list *number*

Configures an entry in an extended XNS access list. The argument *number* is an access list number in the range 500 and 599. The keyword **deny** or **permit** specifies the filtering action. The argument *XNS-protocol* is the XNS protocol number. The NO ACCESS-LIST *number* command *removes the specified access list.*

[**no**] **access-list** *number* {**deny** | **permit**} *XNS-source-network.* [*source-address*[*source-mask*]] *XNS-destination-network.* [*destination-address*[*destination-mask*]]

no access-list *number*

Configures an entry in an XNS access list. The argument *number* is an access list number in the range 400 and 499. The keyword **deny** or **permit** specifies the filtering action. The arguments *XNS-source-network* and *XNS-destination-network* are the only required addresses. The source and destination masks are optional. The NO ACCESS-LIST *number* command *removes the specified access list.*

[**no**] **xns forward-protocol** *type*

Determines which protocol types will be forwarded when a broadcast is received on an interface that has an XNS helper address set. The argument *type* is a decimal number corresponding to an appropriate XNS protocol.Use the NO XNS FORWARD-PROTOCOL command and the appropriate argument to disable the forwarding of the specified protocol.

[**no**] **xns maximum-paths** *paths*

Sets the maximum number of equal-cost paths the router will use. The default value of *paths* is one. The **no** version of the command restores the default.

[**no**] **xns route** *network host-address*

Adds a static route to the XNS routing table. The argument *network* is the destination XNS network number in decimal. The argument *host-address* is a decimal XNS network number and the hexadecimal host number.The **no** version of the command removes the static route.

[**no**] **xns routing** [*address*]

Enables XNS routing. The optional argument *address* is the XNS address the router is to use. If the argument *address* is omitted, the first Ethernet interface hardware address found is used. The **no** version of the command disables all XNS packet processing.

[**no**] **xns ub-emulation**

Enables Ungermann-Bass Net/One routing. Causes Hello packets and routing updates in Ungermann-Bass format to be sent out through all the interfaces on which XNS is enabled. Also causes the DECbrouter 90 to get its routing information for remote networks from Ungermann-Bass updates. Unless the XNS HEAR-RIP command is issued, the DECbrouter 90 will completely ignore standard RIP updates and will behave like a UB router. The no version of the command restores the router to standard XNS mode.

[**no**] **xns ub-routing**

Exactly equivalent to issuing XNS UB-EMULATION for the system as a whole, as well as XNS HEAR-RIP, XNS FLOOD BROADCAST ALLNETS, and NX FLOOD SPECIFIC ALLNETS for all interfaces on which XNS is enabled.

XNS Interface Subcommand Summary

Following is an alphabetically arranged list of the XNS interface subcommands. These commands follow an INTERFACE command.

[**no**] **xns access-group** *number*

Assigns an access list number to an interface and causes all XNS packets that would ordinarily have been forwarded by the router through this interface to be compared to the access list. If a packet fails the access list, it is not transmitted, but is discarded instead. The argument *number* specifies the access list number.The **no** version of the command disables this feature.

[**no**] **xns flood broadcast allnets**

Configures an interface on a router so that any packets received with this destination address will be flooded with packets of destinations -1.FFFF.FFFF.FFFF. The **no** version of the command disables this feature.

[**no**] **xns flood broadcast net-zero**

Configures an interface on a router so that any packets received with this destination address will be flooded with packets of destinations 0.FFFF.FFFF.FFFF. The **no** version of the command disables this feature.

[**no**] **xns flood specific allnets**

Configures an interface on a router so that any packets received with this destination address will be flooded with packets of destinations -1.*specific-host*.The **no** version of the command disables this feature.

[**no**] **xns hear-rip** [*access-list*]

Causes the DECbrouter 90 in Ungermann-Bass mode to receive RIP updates on some interfaces as if they were Ungermann-Bass updates. The argument *access-list* specifies an XNS access list when you only want certain routes to be learned through standard RIP. The DECbrouter 90 will learn from RIP packets only routes to networks permitted by the access list. The **no** version of the command disables this feature.

[**no**] **xns helper-address** *host-address*

Sets a helper address to forward broadcasts. The argument *host-address* is a dotted combination of the network and host addresses. The **no** version of the command disables XNS helpering.

[**no**] **xns input-network-filter** *access-list-number*

Controls which networks are added to your router's routing table. The argument *access-list-number* is the access list number specified in the XNS ACCESS-LIST command.The **no** version of the command disables this feature.

[**no**] **xns network** *number*

Assigns a decimal XNS network number to an interface. The **no** version of the command deassigns the XNS network number for the interface and disables XNS on that interface.

[**no**] **xns output-network-filter** *access-list-number*

Controls the list of networks that are sent out in routing updates by your router. The argument *access-list-number* is the access list number specified in the XNS ACCESS-LIST command. The **no** version of the command disables this feature.

[**no**] **xns route-cache**

Enables fast switching for XNS packets outgoing on this interface. The **no** version of the command disables fast switching.

[**no**] **xns router-filter** *access-list-number*

Controls the list of routers from which data will be accepted. The argument *access-list-number* is the access list number specified in the XNS ACCESS-LIST command. The **no** version of the command disables this feature.

xns update-time *seconds*

Sets the XNS routing update timers to the value assigned to the *seconds* argument. The default is 30 seconds.

7 Configuring Transparent Bridging

This chapter describes the transparent bridging support available in the DECbrouter 90. Topics described in this chapter include:

- An overview of bridging and the DECbrouter 90's implementation of transparent bridging
- Support of source-route transparent (SRT) bridging on Token Ring interfaces
- How to define a spanning tree
- How to adjust the spanning-tree parameters
- How to filter packets based on type; supported types include MAC address, protocol type, vendor code, and LAT multicast service announcements

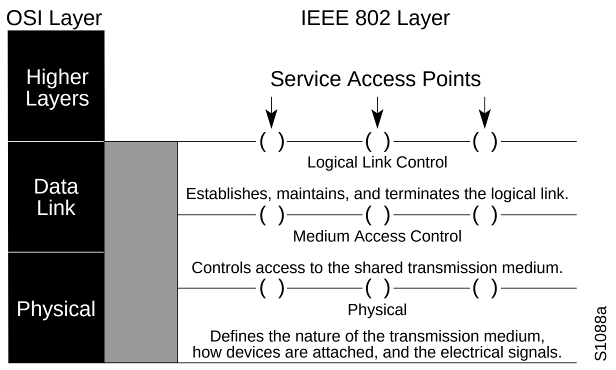
This chapter also provides bridging configuration examples and describes how to maintain, monitor, and debug your transparent bridges. Command summaries are included at the end of the chapter.

Bridging Overview

The basic function of a bridge is to accept frames of data passing through a network, and then make a decision about whether to forward each frame based on information contained in the frame.

Bridges operate at the physical (Level 1) and data link (Level 2) layers of the Open Systems Interconnection (OSI) model. As defined by the IEEE 802.1 standard, the task of the physical layer is to provide the physical connection to the transmission medium. The task of the data link layer is to provide reliable transmission. To accomplish this effectively, the standards committee sublayered the data link layer into two distinct tasks: Logical Link Control (LLC) and Media Access Control (MAC). Bridges, and specifically transparent bridges, function at the physical and MAC layers. However, the administrative filtering functions described later in this chapter use LLC. Figure 7–1 illustrates these concepts.

Figure 7–1 OSI Model and IEEE 802 Layers


OSI Layer
IEEE 802 Layer
Higher Layers
Service Access Points
Data Link
Logical Link Control
Establishes, maintains, and terminates the logical link.
Medium Access Control
Controls access to the shared transmission medium.
Physical
Physical
Defines the nature of the transmission medium, how devices are attached, and the electrical signals.
S1088a

The LLC, as its name implies, controls the logical link; that is, it serves to open, maintain, and terminate a link. It depends upon the physical layers to perform its tasks.

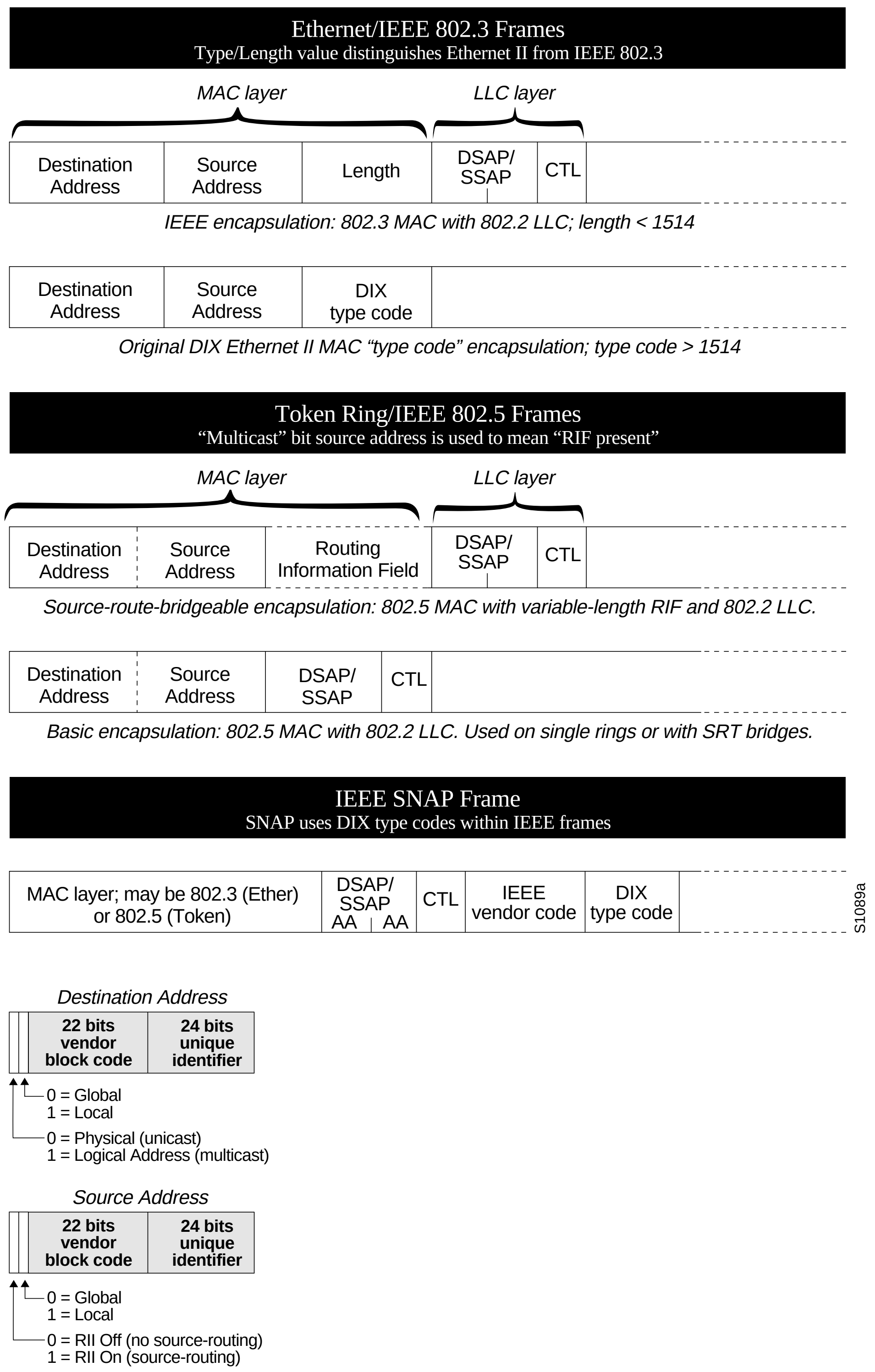
The MAC controls access to the media used to transmit the frames. It acts independently of the physical layer and provides a service interface to the upper layers of the network model. IEEE 802.3 and Ethernet use the same MAC layer, allowing both types of frames to coexist on the same cable. In this way, the same bridge can forward both Ethernet and IEEE 802.3 frames.

The IEEE 802 standards identify two basic elements to facilitate these tasks: a 48-bit address that is either an address for an individual station or a group of stations, and Service Access Points (SAPs), which can be thought of as ports through which a higher-layer application can communicate with its counterparts (peers) operating on other systems.

The IEEE 802 standards also allow any IEEE 802-compliant station to communicate with remote, bridged stations as if they were local. Although the networks connected by a bridge remain physically separate, they appear as one network to the rest of the devices on the internetwork.

Figure 7–2 illustrates the frame formats found on Ethernets.

Figure 7–2 IEEE 802 Frame Formats


Ethernet/IEEE 802.3 Frames
Type/Length value distinguishes Ethernet II from IEEE 802.3
MAC layer
LLC layer
Destination Address
Source Address
Length
DSAP/ SSAP
CTL
IEEE encapsulation: 802.3 MAC with 802.2 LLC; length < 1514
Destination Address
Source Address
DIX type code
Original DIX Ethernet II MAC "type code" encapsulation; type code > 1514
Token Ring/IEEE 802.5 Frames
"Multicast" bit source address is used to mean "RIF present"
MAC layer
LLC layer
Destination Address
Source Address
Routing Information Field
DSAP/ SSAP
CTL
Source-route-bridgeable encapsulation: 802.5 MAC with variable-length RIF and 802.2 LLC.
Destination Address
Source Address
DSAP/ SSAP
CTL
Basic encapsulation: 802.5 MAC with 802.2 LLC. Used on single rings or with SRT bridges.
IEEE SNAP Frame
SNAP uses DIX type codes within IEEE frames
MAC layer; may be 802.3 (Ether) or 802.5 (Token)
DSAP/ SSAP AA AA
CTL
IEEE vendor code
DIX type code
S1089a
Destination Address
22 bits vendor block code
24 bits unique identifier
0 = Global
1 = Local
0 = Physical (unicast)
1 = Logical Address (multicast)
Source Address
22 bits vendor block code
24 bits unique identifier
0 = Global
1 = Local
0 = RII Off (no source-routing)
1 = RII On (source-routing)

Transparent bridges forward frames between networks using the same medium (two IEEE 802.3 networks, for example).

Because bridges examine each frame, and because each frame contains a source address, bridges can easily learn the locations of stations on the network. However, if an address is unknown, the bridges' flooding behavior could also result in endless loops on the network, because all bridges that do not know the address will flood the frame looking for an address. To circumvent this, a tree-type algorithm has been developed that prevents loops and forces the frames into logical routes. Called a spanning-tree algorithm, it allows only one route between any two physical cables or networks.

Spanning-tree bridges are based upon the concept of a *root* bridge that exchanges topology information with designated bridges to maintain the configuration. The root and designated bridges notify all other bridges in the network when topology changes are required, thereby preventing loops and providing a measure of defense against link failure.

Bridges can be further classified as *local* or *remote*. A local bridge typically connects two nearby LANs; for example, Ethernet-to-Ethernet. A remote bridge, on the other hand, links geographically distant LANs or LANs separated by some other media. Two Ethernets linked by a serial line or two Token Rings linked by a frame relay network are examples of remote bridging.

DECbrouter 90's Implementation of Transparent Bridging

The DECbrouter 90's transparent bridging software implementation provides these features:

- Complies with the IEEE 802 standard.
- Provides two spanning-tree algorithms—an older version that is compatible with Digital and other LAN bridges for backwards compatibility and a recent IEEE standard implementation that provides for multiple domains for spanning trees. The software also provides for adjustments to the spanning-tree parameters.
- Allows configuration of filters to effectively block the passing of frames based on Ethernet address, protocol type, or the vendor code. Additionally, the bridging software can be configured to selectively include or exclude LAT multicast service announcements.
- Provides load balancing and redundancy by assigning a set of serial lines between two bridges to a *circuit group*.
- Provides the ability to bridge over X.25 and frame-relay networks.
- Provides for compression of LAT frames to reduce LAT traffic through the network.

Additionally, the DECbrouter 90 router/bridge combines the advantages of a spanning-tree bridge and a full multiprotocol router. This combination provides the speed and protocol transparency of an adaptive spanning-tree bridge, along with the functionality, reliability, and security of a router.

Network interfaces on the interface cards can be configured to serve as both multiprotocol routers and MAC-level bridges, bridging any traffic that cannot otherwise be routed. For example, a router/bridge routing the Internet Protocol can also bridge Digital's Local Area Transport (LAT) protocol or NetBIOS traffic.

Remote bridging over synchronous serial lines also is supported between DECbrouter 90s. As with frames received on all other media types, the learning process and any filtering is applied to frames received on serial lines.

Configuring Transparent Bridging

Follow these steps to configure transparent bridging on a your router/bridge. Each task and command is described in greater detail in the following sections.

1. Define the spanning-tree protocol and assign a bridge group number using the BRIDGE *group* PROTOCOL command. The bridging software supports the IEEE 802.1 spanning-tree protocol and the Digital LANbridge 100 protocol upon which the IEEE standard is based. Additionally, multiple domains are supported for the IEEE spanning tree.
2. Enable each interface and configure any routed protocols. If you want to bridge IP, you must disable IP routing.
3. Assign the network interfaces to the spanning-tree group.
4. Adjust the spanning-tree parameters, as necessary.

The bridging software also supports frame filtering. Filtering can be done by MAC address, protocol type, vendor code, and LAT group codes. Additionally, EXEC-level commands for monitoring and debugging the bridge are available.

Defining the Spanning-Tree Protocol

The router/bridges support two spanning-tree protocols: the IEEE 802.1 standard and the Digital spanning-tree protocol upon which the IEEE standard is based.

To define a spanning-tree protocol, use the BRIDGE *group* PROTOCOL global configuration command. The command has this syntax:

bridge *group* **protocol** {**ieee** | **dec**}
no bridge *group* **protocol** {**ieee** | **dec**}

The argument *group* is a number between one and nine that you choose to refer to a particular set of bridged interfaces. Frames are bridged only among interfaces in the same group. You will use the group number you assign in subsequent bridge configuration commands.

The keyword **protocol** is used to specify one of two spanning-tree protocol types: **ieee** for the Ethernet spanning-tree protocol or **dec** for the Digital LANbridge 100 spanning-tree protocol.

The NO BRIDGE command with the appropriate keywords and arguments deletes the bridge group.

Example

This command defines bridge 1 as using the DECnet spanning-tree protocol.

```
!
bridge 1 protocol dec
!
```

Configuring Transparent Bridging

Establishing Multiple Spanning-Tree Domains

The Digital IEEE 802 bridging software supports multiple spanning-tree domains. You can place any number of router/bridges within the domain. The devices in the domain, and only those devices, will then share spanning-tree information.

Use this feature when multiple routers share the same cable, and you wish to use only certain discrete subsets of those routers to share spanning-tree information with each other. This function is most useful when running other router applications, such as IP UDP flooding, that use the IEEE spanning tree. It also can be used to reduce the number of global reconfigurations in large bridged networks.

Note

Use multiple spanning-tree domains with care. Because bridges in different domains do not share spanning-tree information, bridge loops can be created if the domains are not carefully planned.

Establish a domain by assigning it a value between one and ten using this variation of the BRIDGE global configuration command:

bridge *group* **domain** *domain-number*
no bridge *group* **domain**

The argument *group* is the bridge group number, and must be the same as that specified by the **protocol ieee** keywords in the previously described BRIDGE *group* command.

Use the **domain** keyword to specify a domain; the argument *domain-number* is a domain number you choose. The default domain number is zero; this is the domain number required when communicating to IEEE bridges that do not support this domain extension. The no version of the command returns it to a single bridge domain by choosing domain zero (0).

Note

This command works only when the bridge group is running the IEEE spanning-tree protocol.

Example

This command places bridge group 1 in bridging domain 3. Only other routers that are in domain 3 will accept spanning-tree information from this router.

```
!
bridge 1 domain 3
!
```

Configuring Transparent Bridging

Bridging or Routing IP

All protocols except IP are bridged by a router/bridge unless their routing is explicitly enabled. Refer to the *DECbrouter 90 Products Configuration and Reference, Volume 2* for the procedures to enable routing of individual protocols. IP is normally routed by the router/bridge.

Also note that bridging and routing are done on a per-system basis. If a protocol is being routed, it must be routed on all interfaces that are handling that protocol. This is similar for bridging. You cannot route IP on one interface and bridge it on another interface.

To bridge IP you must disable IP routing by giving the following global configuration command:

no ip routing

Assign the *same* IP address to all network interfaces to manage the system with Telnet, TFTP, SNMP, ICMP (ping), and so forth. Once bridging is enabled, all IP and ARP frames not intended for the router/bridge are handled according to standard bridging and spanning-tree rules. IP routing processes such as IGRP or RIP must not be running.

Assigning Each Interface to a Spanning-Tree Group

Assign each network interface to a spanning-tree group using the BRIDGE-GROUP *group* interface subcommand, which has this syntax:

bridge-group *group* [**cbus-flooding**]
no bridge-group *group* [**cbus-flooding**]

The simplest form of the BRIDGE-GROUP command takes just a spanning-tree group number as an argument. More complex forms of the BRIDGE-GROUP command are described later. The **no** version of the command removes the interface from the bridge group.

Note

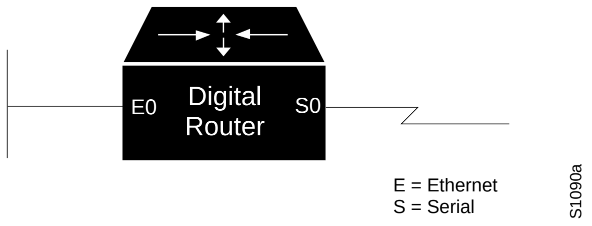
Serial interfaces must be running with HDLC, X.25, or frame relay encapsulation in order to support bridging.

Basic Bridging Example

Figure 7–3 is an example of a basic bridging configuration. The system has one Ethernet, and one serial line. The Internet Protocol (IP) is being routed, and everything else is being bridged. The DEC-compatible bridging algorithm with default parameters is being used.

Configuring Transparent Bridging

Figure 7–3 Example of Basic Bridging



The configuration file for the router/bridge depicted in Figure 7–3 would be as follows:

```
!
interface ethernet 0
ip address 192.31.7.26 255.255.255.240
bridge-group 1
!
interface serial 0
ip address 192.31.7.34 255.255.255.240
bridge-group 1
!
bridge 1 protocol dec
```

Adjusting Spanning-Tree Parameters

You may need to adjust certain spanning-tree parameters. Parameters affecting the entire spanning tree are configured with variations of the BRIDGE *group* global configuration command. Interface-specific parameters are configured with variations of the BRIDGE-GROUP *group* interface subcommand.

Note

Only network administrators with a good understanding of how bridges and the spanning-tree protocol work should make adjustments to spanning-tree parameters. Poorly planned adjustments to these parameters can have a negative impact on performance. A good source on bridging is the IEEE 802.1d specification; see the recommended reading list in Appendix F for other references.

Electing the Root Bridge

Configure the priority of an individual bridge or the likelihood that it will be selected as the root bridge, with the BRIDGE *group* PRIORITY global configuration command. The command has this syntax:

bridge *group* **priority** *number*

The argument *group* is the bridge group number. The argument *number* can range from 1 to 65000. The lower the value of *number*, the more likely the bridge will be chosen as root. The default priority is 128.

When two bridges tie for position as the root bridge, an interface priority determines which bridge will serve as the root bridge. Use the BRIDGE-GROUP *group* PRIORITY *number* interface subcommand to control this.

Example

This command establishes this bridge as a likely candidate to be the root bridge.

```
!
bridge 1 priority 100
!
```

Adjusting the Interval Between Hello BPDUs

The interval between Hello Bridge Protocol Data Units (BPDUs) is specified with the BRIDGE *group* HELLO-TIME global configuration command. The command has this syntax:

bridge *group* **hello-time** *seconds*

The argument *group* is the bridge group number. The argument *seconds* can be any value between one and ten seconds. The default value is one second.

Example

This command sets the interval to five seconds.

```
!
bridge 1 hello-time 5
!
```

Note

Each bridge in a spanning tree adopts the **hello-time, forward-time,** and **max-age** parameters of the root bridge, regardless of what its individual configuration might be.

Defining the Forward Delay Interval

The forward delay interval is the amount of time spent listening for topology change information after an interface has been activated for bridging and before forwarding actually begins. Use the BRIDGE *group* FORWARD-TIME global configuration command to specify this interval. The command has this syntax:

bridge *group* **forward-time** *seconds*

The argument *group* is the bridge group number. The argument *seconds* can be any value between 10 and 200 seconds. The default value is 30 seconds.

Note

Each bridge in a spanning tree adopts the **hello-time, forward-time,** and **max-age** parameters of the root bridge, regardless of what its individual configuration might be.

Example

This command sets the forward delay interval to 60 seconds.

```
!
bridge 1 forward-time 60
!
```

Defining the Maximum Idle Interval

If a bridge does not hear BPDUs from the root bridge within a specified interval, it assumes that the network has changed and recomputes the spanning-tree topology. This interval is 15 seconds by default. It can be changed with the BRIDGE *group* MAX-AGE global configuration command. The command has this syntax:

bridge *group* **max-age** *seconds*

The argument *group* is the bridge group number. The argument *seconds* is the interval the bridge will wait to hear BPDUs from the root bridge.

Note

Each bridge in a spanning tree adopts the hello-time, forward-time, and max-age parameters of the root bridge, regardless of what its individual configuration might be.

Example

This command increases the maximum idle interval to 20 seconds.

```
!
bridge 1 max-age 20
!
```

Assigning Path Costs

Each interface has a path cost associated with it. By convention, the path cost is 1000/data rate of the attached LAN, in Mbps. Ethernet path cost is 100 Mbps.

Use the interface subcommand BRIDGE-GROUP *group* PATH-COST to set a different path cost. The command syntax is as follows:

bridge-group *group* **path-cost** *cost*
no bridge-group *group* **path-cost** *cost*

The argument *group* is the bridge group number. The path cost can range from 0 to 65535, with higher values indicating higher costs. The default path cost is computed from the interface's bandwidth setting. The **no** version of the command chooses the default path cost for the interface.

Example

This command changes the default path cost for interface Ethernet 0.

```
!
interface ethernet 0
bridge-group 1 path cost 250
!
```

Setting an Interface Priority

A priority also can be set for an interface. When two bridges tie for position as the root bridge, an interface priority is used to break the tie. Use the BRIDGE-GROUP *group* PRIORITY interface subcommand to do so. The command has this syntax:

bridge-group *group* **priority** *number*

The argument *group* is the bridge group number. The argument *number* can range from 0 to 255. The default value is zero, and the lower the number, the more likely it is that the bridge on this interface will be chosen as the root.

Example

This command increases the likelihood that the root bridge will be the one on Ethernet 0 in spanning-tree group 1.

```
!
interface ethernet 0
bridge-group 1 priority 0
!
```

Establishing Administrative Filtering

A bridge examines frames and transmits them through the internetwork according to the destination address; a bridge will not forward a frame back to its originating network segment. The bridge software allows specific administrative filters to be configured that block frames based upon information other than paths to their destinations. This administrative filtering can be done by one of the following methods:

- MAC-layer address
- Protocol type—Ethernet, IEEE 802 LLC, or IEEE 802 SNAP
- Vendor code
- LAT multicast service announcement
- Extended access lists

When setting up administrative filtering, remember that there is virtually no performance penalty in filtering by MAC address or vendor code, but there can be a significant performance penalty when filtering by protocol type.

Administrative Filtering by MAC Layer Address

Blocking transmission of frames based on MAC layer address is configured by variations of the BRIDGE *group* global configuration command, as described in the following sections.

Filtering on MAC Address

To filter frames with a particular MAC layer station source or destination address, use the global configuration command BRIDGE *group* ADDRESS. The full syntax of this command follows.

bridge *group* **address** *MAC-address* [**forward** | **discard**] [*interface*]
no bridge *group* **address** *MAC-address*

The argument *group* is the group number you assigned to the spanning tree.

The argument *MAC-address* is a 48-bit dotted-triple hardware address such as that displayed by the EXEC SHOW ARP command. It is either a station address, the broadcast address, or a multicast destination address.

An example is:

```
0800.cb00.45e9
```

The optional keywords **forward** and **discard** are described in Table 7–1.

Table 7–1 Optional MAC Address Filtering Keywords

Keyword	Function
forward	A frame sent from or destined to the specified address is forwarded as appropriate.
discard	A frame sent from or destined to the specified address is discarded without further processing.

The argument *interface* is an optional interface specification, such as Ethernet 0, and is added after the **discard** or **forward** keyword to indicate the interface on which that address can be reached.

Any number of addresses can be configured into the system without a performance penalty.

Use the NO BRIDGE *group* ADDRESS command followed by the MAC address to disable the forwarding ability.

Note

MAC addresses on Ethernets are "bit swapped" when compared with MAC addresses on Token Ring and FDDI. For example, address 0110.2222.3333 on Ethernet is 8008.4444.CCCC on Token Ring and FDDI. Access lists always use the canonical Ethernet representation. When using different media and building access lists to filter on MAC addresses, keep this point in mind. Note that when a bridged packet traverses a serial link, it has an Ethernet-style address.

Examples

This command enables frame filtering with MAC address 0800.cb00.45e9.

```
!
bridge 1 address 0800.cb00.45e9 forward ethernet 1
!
```

The frame is forwarded through interface Ethernet 1.

This command disables the ability to forward frames with MAC address 0800.cb00.45e9.

```
!
no bridge 1 address 0800.cb00.45e9
!
```

Preventing the Forwarding of Dynamically Determined Stations

Normally, the system forwards any frames for stations that it has learned about dynamically. This behavior can be changed with the NO BRIDGE *group* ACQUIRE global configuration command. The full syntax of this command follows:

no bridge *group* **acquire**
bridge *group* **acquire**

The argument *group* is the bridge group number. The bridge filters out all frames except those whose sourced-by or destined-to addresses have been statically configured into the forwarding cache.

The BRIDGE *group* ACQUIRE global configuration command restores the default behavior.

Example

This command prevents the forwarding of dynamically determined source and destination addresses.

```
!
no bridge 1 acquire
!
```

Forwarding the Multicast Addresses

The bridging support can be configured to allow the forwarding, but not the learning, of frames received with multicast source addresses. Use this variation of the BRIDGE *group* global configuration command to configure this function:

bridge *group* **multicast-source**
no bridge *group* **multicast-source**

List the bridge group using the *group* argument.The **no** version of the command disables this function on the bridge.

Administrative Filtering by Protocol Type

Filtering by protocol type is done using the access list mechanism and by specifying a protocol type code. The bridge ACCESS-LIST command specifies an element in an access list. The order in which ACCESS-LIST commands are entered affects the order in which the access conditions are checked. Each condition is tested in succession. A matching condition is then used to execute a permit or deny decision. If no conditions match, a deny decision is reached.

Establishing Protocol Type Access Lists

Type-code access lists are built with this bridge ACCESS-LIST global configuration command:

access-list *list* {**permit** | **deny**} *type-code wild-mask*
no access-list *list*

The argument *list* is a user-selectable number between 200 and 299 that identifies the list.

The keyword **permit** permits the frame; the keyword **deny** denies the frame.

The argument *type-code* is a 16-bit hexadecimal number written with a leading "0x"; for example, 0x6000. You can specify either an Ethernet type code for Ethernet-encapsulated packets or a DSAP/SSAP pair for 802.3 or 802.5-encapsulated packets. Ethernet type codes are listed in Appendix C of this manual.

The argument *wild-mask* is another 16-bit hexadecimal number whose ones bits correspond to bits in the *type-code* argument that should be ignored when making a comparison. (A mask for a DSAP/SSAP pair should always be at least 0x0101. This is because these two bits are used for purposes other than identifying the SAP codes.)

The **no** version of the command removes a single access list entry.

Examples

The following access list permits only LAT frames (type 0x6004) and filters out all other frame types:

```
!
access-list 201 permit 0x6004 0x0000
!
```

The following access list filters out only type codes assigned to Digital (0x6000 through 0x600F) and lets all other types pass:

```
!
access-list 202 deny   0x6000 0x000F
access-list 202 permit  0x0000 0xFFFF
!
```

Use the last item of an access list to specify a default action; for example, permit everything else or deny everything else. If nothing else in the access matches, the default action is normally to deny access; that is, filter out all other type codes.

Note

Type-code access lists can have an impact on system performance; therefore, keep the lists as short as possible and use wildcard bit masks whenever possible.

Filtering Ethernet- and SNAP-Encapsulated Packets on Input

You can filter Ethernet- and SNAP-encapsulated packets on input. For SNAP-encapsulated frames, the list is applied against the two-byte TYPE field given after the DSAP/SSAP/OUI fields in the frame.

The access list specifying the type codes to be filtered is given in this variation of the BRIDGE-GROUP interface subcommand:

bridge-group *group* **input-type-list** *list*

The argument *group* is the spanning-tree group number.

The argument *list* is the access list number you assigned with the bridge ACCESS-LIST command. Specify a 0 for *list* to disable the application of the access list on the bridge group.

This access list is then applied to all Ethernet and SNAP frames received on that interface prior to the bridge learning process. SNAP frames also must pass any applicable IEEE 802 DSAP/SSAP access lists.

Example

The following example illustrates how to configure a system with one Ethernet interfaces, and two serial interfaces. The system is routing both IP and DECnet. Each interface has an access list that allows only the LAT protocol to be bridged. The bridging software also has been instructed to discard frames sent to or from the address AB00.0C00.AE35.

```
!
decnet address 34.88
!
interface ethernet 0
ip address 192.31.7.26 255.255.255.240
decnet cost 10
bridge-group 1
bridge-group 1 input-type-list 201
!
interface serial 0
ip address 192.31.7.34 255.255.255.240
decnet cost 10
bridge-group 1
bridge-group 1 input-type-list 201
!
interface serial 1
ip address 192.31.7.65 255.255.255.240
decnet cost 10
bridge-group 1
bridge-group 1 input-type-list 201
!
bridge 1 protocol dec
bridge 1 address AB00.0C00.AE35 discard
!
access-list 201 permit 0x6004 0x0000
access-list 201 deny  0x0000 0xFFFF
!
```

Filtering Ethernet- and SNAP-Encapsulated Packets on Output

You can filter Ethernet- and SNAP-encapsulated packets on output. The access specifying the type codes to be filtered is given by this variation of the BRIDGE-GROUP interface subcommand:

bridge-group *group* **output-type-list** *list*

The argument *group* is the spanning-tree group number.

The argument *list* is the access list number you assigned with the bridge ACCESS-LIST command. Specify a zero (0) for *list* to disable the application of the access list on the bridge group.

This access list is applied just before sending out a frame to an interface.

Example

This command specifies access list 202 on interface Ethernet 0.

```
!
interface ethernet 0
bridge-group 2 output-type-list 202
!
```

Filtering IEEE 802-Encapsulated Packets on Input

You can filter IEEE 802-encapsulated packets on input. The access list specifying the type codes to be filtered is given by this variation of the BRIDGE-GROUP interface subcommand:

bridge-group *group* **input-lsap-list** *list*

The argument *group* is the spanning-tree group number.

The argument *list* is the access list number you assigned with the bridge ACCESS-LIST command. Specify a zero (0) for *list* to disable the application of the access list on the bridge group.

This access list is applied to all IEEE 802 frames received on that interface prior to the bridge-learning process. SNAP frames also must pass any applicable Ethernet type-code access list.

Example

This command specifies access list 203 on interface Ethernet 0:

```
!
interface ethernet 0
bridge-group 3 input-lsap-list 203
!
```

Filtering IEEE 802-Encapsulated Packets on Output

You can filter IEEE 802-encapsulated packets on output. The access list specifying the type codes to be filtered is given by this variation of the BRIDGE-GROUP interface subcommand:

bridge-group *group* **output-lsap-list** *list*

The argument *group* is the spanning-tree group number.

The argument *list* is the access list number you assigned with the bridge ACCESS-LIST command. Specify a zero (0) for *list* to disable the application of the access list on the bridge group.

SNAP frames also must pass any applicable Ethernet type-code access list. This access list is applied just before sending out a frame to an interface.

Note

For performance reasons, it is not a good idea to have both input and output type code filtering on the same interface.

Note

Access lists for Ethernet- and IEEE 802-encapsulated packets affect only bridging functions. It is not possible to use such access lists to block frames with protocols that are being routed.

Example

This command specifies access list 204 on interface Ethernet 0.

```
!
interface ethernet 0
bridge-group 4 output-lsap-list 204
!
```

Administrative Filtering by Vendor Code

The bridging software supports administrative filtering of MAC addresses. These lists support filtering groups of MAC addresses, including those with particular vendor codes. The lists are defined with the bridge ACCESS-LIST global configuration command and the BRIDGE-GROUP interface subcommand. There is no noticeable performance loss in using these access lists. The lists can be of indefinite length. The following sections describe how to set up these lists.

Note

Remember that here, as with any access list using MAC addresses, Ethernets swap their MAC address bit ordering, and Token Rings and FDDI do not. As such, an access list that works for one medium may not work for others.

Establishing Vendor Code Access Lists

Use the bridge ACCESS-LIST global configuration command to establish MAC address access lists. The command has the following form:

access-list *list* {**permit** | **deny**} *address mask*
no access-list *list*

The argument *list* is an integer from 700 to 799 that you select for the list.

The argument *address* and mask are 48-bit MAC addresses written in dotted triplet form. The ones bits in the *mask* argument are the bits to be ignored in *address*.

See the next section Filtering Source Addresses for an example of use of this command. The **no** version of the command removes a single access list entry.

Filtering Source Addresses

Use the BRIDGE-GROUP *group* INPUT-ADDRESS-LIST interface subcommand to assign an access list to a particular interface for filtering on the MAC source addresses of packets received on that interface. The command has this syntax:

bridge-group *group* **input-address-list** *list*
no bridge-group *group* **input-address-list** *list*

The argument *group* is the spanning-tree group number.

The argument *list* is the access list number you assigned with the bridge ACCESS-LIST command.

Example

This configuration example assumes you want to disallow the bridging of Ethernet packets of all Sun workstations on Ethernet 0. Software assumes that all such hosts have Ethernet addresses with the vendor code 0800.2000.0000. The first line of the access list denies access to all Sun workstations, while the second line permits everything else. You then assign the access list to the input side of Ethernet 0.

```
!
access-list 700 deny 0800.2000.0000 0000.00FF.FFFF
access-list 700 permit 0000.0000.0000 FFFF.FFFF.FFFF
interface ethernet 0
bridge-group 1 input-address-list 700
!
```

Filtering Destination Addresses

Use the BRIDGE-GROUP *group* OUTPUT-ADDRESS-LIST interface subcommand to assign an access list to a particular interface for filtering the MAC destination addresses of packets that would ordinarily be forwarded out that interface. The command has this syntax:

bridge-group *group* **output-address-list** *list*
no bridge-group *group* **output-address-list** *list*

The argument *group* is the spanning-tree group number.

The argument *list* is the access list number you assigned with the bridge ACCESS-LIST command.

Example

This command assigns access list 703 to interface Serial 0.

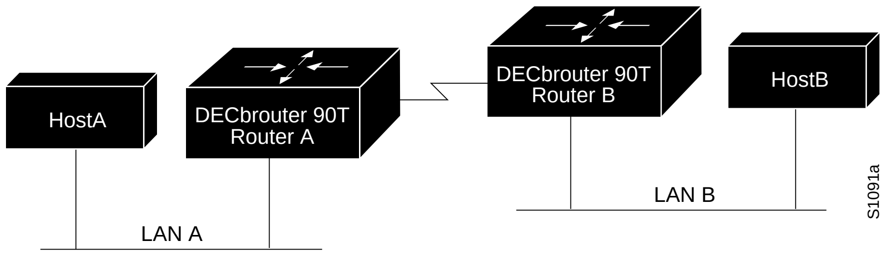
```
!
interface serial 0
bridge-group 5 output-address-list 703
!
```

Accommodating Multicast or Broadcast Packets

When filtering specific MAC destination addresses, allow for multicast or broadcast packets that are required by the bridged network protocols.

Assume you are bridging IP in the network illustrated in Figure 7–4.

Figure 7–4 Network Demonstrating Output Address List Filtering



The MAC address of HostA is 0800.0907.0207, and the MAC address of HostB is 0260.8c34.0864. The following configuration would work as expected, because input addresses work on source address on the incoming interface:

```
access-list 700 permit 0260.8c34.0864 0000.0000.0000
access-list 700 deny 0000.0000.0000 FFFF.FFFF.FFFF
interface ethernet 0
bridge-group 1 input-address-list 700
```

However, the following configuration may work initially, but will eventually fail, even though the destination address on the output interface is correct:

```
access-list 700 permit 0260.8c34.0864 0000.0000.0000
access-list 700 deny 0000.0000.0000 FFFF.FFFF.FFFF
interface ethernet 0
bridge-group 1 output-address-list 700
```

The preceding configuration does not allow for an ARP broadcast with a destination address of FFFF.FFFF.FFFF, so the access list fails. The correct access list would be as follows:

```
access-list 700 permit 0260.8c34.0864 0000.0000.0000
access-list 700 permit FFFF.FFFF.FFFF 0000.0000.0000
access-list 700 deny 0000.0000.0000 FFFF.FFFF.FFFF
interface ethernet 0
bridge-group 1 output-address-list 700
```

Administrative Filtering of LAT Service Announcements

The bridging software provides LAT frame filtering. LAT bridge filtering allows the selective inclusion or exclusion of LAT multicast service announcements, on a per-interface basis.

In the LAT protocol, a *group code* is defined as a decimal number in the range 0 to 255. Some of the LAT configuration commands take a list of group codes; this is referred to as a *group code list*. The rules for entering numbers in a group code list follow.

- Entries can be individual group code numbers separated with a space. (The Digital LAT implementation specifies that a list of numbers be separated by commas; however, the DECbrouter 90's implementation expects the list to be separated by a space.)
- Entries also can specify a range of numbers. This is done by separating an ascending order range of group numbers with a hyphen.

- Any number of group codes or group code ranges can be listed in one command; just separate each with a space.

In LAT, each node transmits a periodic service advertisement message that announces its existence and availability for connections. Within the message is a group code list; this is a mask of up to 256 bits. Each bit represents a group number.

A terminal server only will connect to a host system when there is an overlap between the group code list of the user on the terminal server and the group code list in the service advertisement message.

In an environment with many bridges and many LAT hosts, the number of multicast messages that each system has to deal with becomes significant. The 256 group codes may not be enough to allocate local assignment policies, such as giving each DECserver 200 device its own group code, in large bridged networks.

LAT group code filtering allows you to have very fine control over which multicast messages actually get bridged. Through a combination of input and output permit and deny lists, many different LAT control policies can be implemented. Use the EXEC command SHOW SPAN to report the group code filtering in effect.

Specifying LAT Group Code Service Filtering

Use the BRIDGE *group* LAT-SERVICE-FILTERING global configuration command to specify LAT group-code filtering. The command has this syntax:

bridge *group* **lat-service-filtering**
no bridge *group* **lat-service-filtering**

The command informs the system that LAT service advertisements require special processing. Use the *group* argument to specify the bridge group in which this special processing is to take place.

The **no** version of this command disables the use of LAT service filtering on the bridge group.

Example

This command specifies that LAT service announcements traveling across bridge group 1 require some special processing.

```
!
bridge 1 lat-service-filtering
!
```

Specifying Deny Conditions for LAT Group Codes on Input

Use the BRIDGE-GROUP *group* INPUT-LAT-SERVICE-DENY interface subcommand to specify the group codes by which to deny access upon input. The command has this syntax:

bridge-group *group* **input-lat-service-deny** *grouplist*

This command causes the system to not bridge any LAT service advertisement that has any of the specified groups set. The argument *group* is the previously chosen spanning-tree group number. Enter the list of LAT service groups with the *grouplist* argument. Specify a zero (0) to disable the LAT group code for the bridge group.

Example

This command causes any advertisements with groups 6, 8, and 14 through 20 to be dropped.

```
!
interface ethernet 0
bridge-group 1 input-lat-service-deny 6 8 14-20
!
```

Specifying Permit Conditions for LAT Group Codes on Input

Use the BRIDGE-GROUP *group* INPUT-LAT-SERVICE-PERMIT interface subcommand to specify the group codes by which to permit access upon input. The command has this syntax:

bridge-group *group* **input-lat-service-permit** *grouplist*

This command causes the system to bridge only those service advertisements that match at least one group in the group list specified by the *grouplist* argument. Specify a zero (0) to disable the LAT group code for the bridge group.

Note

If a message specifies group codes in both the deny and permit list, the message is not bridged.

Example

This command bridges any advertisements from groups 1, 5, and 12 through 14.

```
!
interface ethernet 1
bridge-group 1 input-lat-service-permit 1 5 12-14
!
```

Specifying Deny Conditions for LAT Group Codes on Output

Use the BRIDGE-GROUP *group* OUTPUT-LAT-SERVICE-DENY interface subcommand to specify the group codes by which to deny access upon output. The command has this syntax:

bridge-group *group* **output-lat-service-deny** *grouplist*
no bridge-group *group* **output-lat-service-deny** *grouplist*

This command causes the system to not bridge onto this output interface any service advertisements that contain groups matching any of these in the group list. The LAT service advertisements are specified with the argument *grouplist*.

The argument *group* is the previously chosen spanning-tree group number. The **no** version of the command disables this application on the bridge group.

Example

This command prevents bridging of LAT service announcements from groups 12 through 20.

```
!
interface ethernet 0
bridge-group 1 output-lat-service-deny 12-20
!
```

Specifying Permit Conditions for LAT Group Codes on Output

Use the BRIDGE-GROUP *group* OUTPUT-LAT-SERVICE-PERMIT interface subcommand to specify the group codes by which to permit access upon output. The command has this syntax:

bridge-group *group* **output-lat-service-permit** *grouplist*
no bridge-group *group* **output-lat-service-permit** *grouplist*

This command causes the system to bridge onto this output interface only those service advertisements that match at least one group in the specified group code list. The service advertisements are specified with the argument *grouplist*.

The argument *group* is the previously chosen spanning-tree group number. The **no** version of the command disables this application on the bridge group.

Note

If a message matches both a deny and a permit condition, it will not be bridged.

Example

This command allows only LAT service announcements from groups 5, 12, and 20 on this bridge.

```
!
interface ethernet 0
bridge-group 1 output-lat-service-permit 5 12 20
!
```

Extended Access Lists for Transparent Bridging

Extended access lists allow finer granularity of control. They allow you to specify both source and destination addresses and arbitrary bytes in the packet.

To define an extended access list, use the extended version of the ACCESS-LIST subcommand.

access-list *list* {**permit** | **deny**} *source source-mask destination destination-mask offset-in-packet size operator operand*

The argument *list* is an integer from 1100 through 1199 that you assign to identify one or more permit/deny conditions as an extended access list. Note that a list number in the range 1100 to 1199 distinguishes an extended access list from other access lists. The condition keywords **permit** and **deny** determine whether the router allows or disallows a connection when a packet matches an access condition. The router stops checking the extended access list after a match occurs. All conditions must be met to make a match.

The argument *source* is a MAC Ethernet address in the form xxxx.xxxx.xxxx. The argument *source-mask* is a mask of MAC Ethernet source address bits to be ignored. The router uses the *source* and *source-mask* arguments to match the source address of a packet. The arguments *destination* and *destination-mask* are MAC Ethernet values used for matching the destination address of a packet.

Using the *offset-into-packet* and the *size* arguments, you can define a range of values that must be satisfied in the access list. The *offset-into-packet* is specified in decimal or in hexadecimal format in the form 0xnn. The size is an integer 1 to 4.

Use the arguments *operator* and *operand* to compare arbitrary bytes within the packet.

The argument *operator* can be one of these keywords:

- **lt**—less than
- **gt**—greater than
- **eq**—equal
- **neq**—not equal
- **and**—bitwise and
- **xor**—bitwise exclusive or

The argument *operand* is a value to be compared to or masked against.

Note

After an access list is created initially, any subsequent additions (possibly entered from the terminal) are placed at the *end* of the list. In other words, you cannot selectively add or remove access lists command lines from a specific access list.

Example

The following example permits packets from MAC addresses 000c.1Bxx.xxxx to any MAC address if the packet contains a value less than 0x55AA in the two bytes that begins 0x1E bytes into the packet.

```
interface ethernet 0
bridge-group 3 output-pattern 1102
access-list 1102 permit 000c.1b00.0000 0000.00ff.ffff
    0000.0000.0000 ffff.ffff.ffff 0x1e 2
    lt 0x55aa
```

Caution

Do not specify offsets into a packet that are greater than the size of the packet.

Special Bridging Configurations

This section describes some special bridging configurations, including how to configure load balancing over serial lines, how to configure X.25 and frame-relay bridging, and how to compress LAT traffic.

Establishing Load Balancing

In the normal operation of the spanning-tree algorithm, parallel network segments cannot all be carrying traffic at the same time. This is necessary to prevent the looping of frames. In the case of serial lines, however, there is often a desire to increase the available bandwidth by using multiple, parallel serial lines.

To modify the spanning-tree algorithm's handling of serial lines, a set of serial lines between two bridges can be grouped in an association called a *circuit group*. If the spanning-tree algorithm allows any serial interface in the circuit group to forward packets, then all interfaces can be used for forwarding traffic. Ordering problems are avoided by assigning each destination address to a particular serial interface. Reassignment is done dynamically if interfaces go down or come back up.

To establish load balancing, use the BRIDGE-GROUP *group* CIRCUIT interface subcommand. The command has this syntax:

bridge-group *group* **circuit** *number*

The command marks a serial interface as belonging to circuit group. The argument group is the spanning-tree group number. The argument *number* defines the circuit group number and is a small integer less than ten. Specify a zero (0) for *number* to disable the circuit group number.

The parallel serial interfaces on both bridges must all be marked as being members of the same circuit group.

Example

To load share over the two parallel serial links in this example, each router would have the configuration shown here.

```
!
interface ethernet 0
bridge-group 1
!
interface serial 0
bridge-group 1
bridge-group 1 circuit 1
!
interface serial 1
bridge-group 1
bridge-group 1 circuit 1
!
bridge 1 protocol dec
!
```

Configuring X.25 Bridging

The transparent bridging software supports bridging of packets in X.25 frames. This ability is useful, as an example, for transmitting packets from proprietary protocols across an X.25 network.

To configure this capability, use this variation of the X25 MAP interface subcommand in the bridging configuration file:

x25 map bridge *X.121-address* **broadcast** [*options-keywords*]
no x25 map bridge

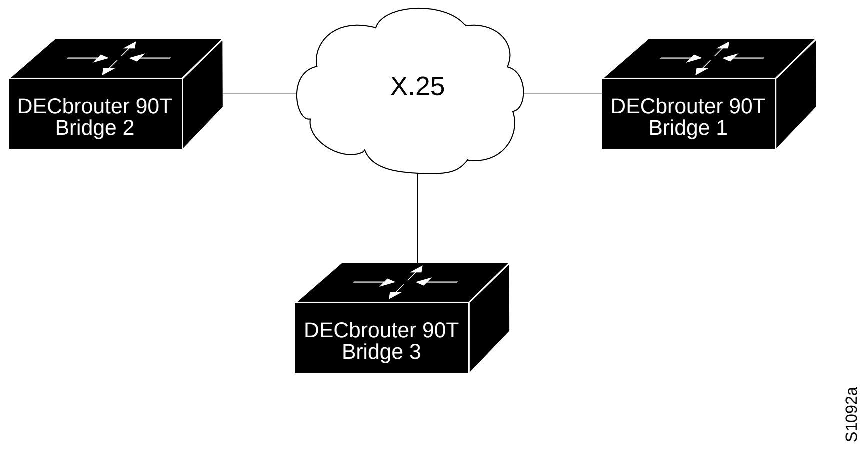
The keyword **bridge** specifies bridging over X.25. The argument *X.121-address* is the X.121 address. The keyword **broadcast** is required for bridging over X.25. The optional argument *options-keywords* is the services that can be added to this map.

The X.25 bridging software uses the same spanning-tree algorithm as the other bridging functions, but allows packets to be encapsulated in X.25 frames and transmitted across X.25 media. The command specifies Internet-to-X.121 address mapping and maintains a table of both the Ethernet and X.121 addresses.

X.25 Bridging Example

Figure 7–5 is an example configuration illustrating three bridges connected to each other through an X.25 network.

Figure 7–5 X.25 Bridging Example



Following are the configuration commands for each of the bridges depicted in Figure 7–5.

Example for Bridge 1

```
interface ethernet 0
bridge-group 5
ip address 128.88.11.9 255.255.255.0
!
interface serial 0
encapsulation x25
x25 address 31370019027
bridge-group 5
x25 map bridge 31370019134 broadcast
x25 map bridge 31370019565 broadcast
!
bridge 5 protocol ieee
!
```

Example for Bridge 2

```
interface serial 1
encapsulation x25
x25 address 31370019134
bridge-group 5
x25 map bridge 31370019027 broadcast
x25 map bridge 31370019565 broadcast
!
bridge 5 protocol ieee
!
```

Example for Bridge 3

```
interface serial 0
encapsulation x25
x25 address 31370019565
bridge-group 5
x25 map bridge 31370019027 broadcast
x25 map bridge 31370019134 broadcast
!
bridge 5 protocol ieee
!
```

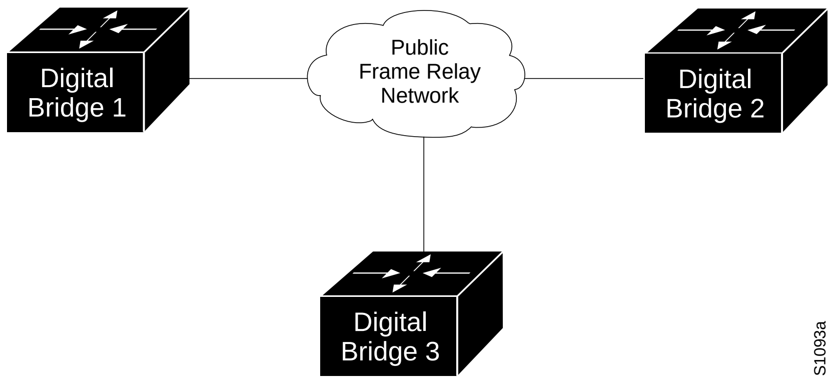
Configuring Frame-Relay Bridging

The transparent bridging software supports bridging of packets over frame relay networks. This ability is useful, as an example, for transmitting packets from proprietary protocols across a frame relay network.

Bridging over a frame relay network is supported both on networks that support a multicast facility and those that do not. Both cases are described in the sections that follow.

Figure 7–6 illustrates three bridges connected to each other through a frame relay network.

Figure 7–6 Frame-Relay Bridging Example



Bridging in a Frame Relay Network with no Multicasts

To configure bridging in a network not supporting a multicast facility, use this variation of the FRAME-RELAY MAP interface subcommand.

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

The keyword **bridge** specifies bridging over frame relay. The argument DLCI is the Data Link Connection Identifier (DLCI) of the destination bridge. The keyword **broadcast** is required for bridging.

The frame relay bridging software uses the same spanning-tree algorithm as the other bridging functions, but allows packets to be encapsulated for transmission across a frame relay network. The command specifies Internet-to-DLCI address mapping and maintains a table of both the Ethernet and DLCIs.

Following are the configuration commands for each of the bridges in a network that does not support a multicast facility.

Example for Bridge 1

```
interface ethernet 0
bridge-group 5
ip address 128.88.11.9 255.255.255.0
!
interface serial 0
encapsulation frame-relay
bridge-group 5
frame-relay map bridge 134 broadcast
frame-relay map bridge 565 broadcast
!
bridge 5 protocol ieee
!
```

Example for Bridge 2

```
interface serial 1
encapsulation frame-relay
bridge-group 5
frame-relay map bridge 27 broadcast
frame-relay map bridge 565 broadcast
!
bridge 5 protocol ieee
!
```

Example for Bridge 3

```
interface serial 0
encapsulation frame-relay
bridge-group 5
frame-relay map bridge 27 broadcast
frame-relay map bridge 134 broadcast
!
bridge 5 protocol ieee
!
```

Bridging in a Frame Relay Network with Multicasts

The multicast facility is used to learn about the other bridges on the network, eliminating the need for the FRAME-RELAY MAP commands.

Following are the configuration commands for each of the bridges in a network that supports a multicast facility.

Example for Bridge 1

```
interface ethernet 2
bridge-group 5
ip address 128.88.11.9 255.255.255.0
!
interface serial 0
encapsulation frame-relay
bridge-group 5
!
bridge 5 protocol ieee
!
```

Example for Bridge 2

```
interface serial 1
encapsulation frame-relay
bridge-group 5
!
bridge 5 protocol ieee
!
```

Example for Bridge 3

```
interface serial 0
encapsulation frame-relay
bridge-group 5
!
bridge 5 protocol ieee
!
```

Configuring LAT Compression

The Local Area Transport (LAT) protocol used by Digital and Digital-compatible terminal servers is one of the common protocols that lacks a well-defined network layer (Level 3), and so always must be bridged.

To reduce the amount of bandwidth that LAT traffic consumes on serial interfaces, you can specify a LAT-specific form of compression. This is done with the BRIDGE-GROUP *group* LAT-COMPRESSION interface subcommand. The command has this syntax:

bridge-group *group* **lat-compression**
no bridge-group *group* **lat-compression**

The argument *group* is the spanning-tree group number.

Compression is applied to LAT frames being sent out the router/bridge through the interface in question.

LAT compression can be specified only for serial interfaces. For the most common LAT operations (user keystrokes and acknowledgment packets), LAT compression reduces LATÕs bandwidth requirements by nearly a factor of two. The **no** version of this command disables LAT compression on the bridge group.

Example

This command compresses LAT frames on the bridge assigned to group 1.

```
!
bridge-group 1 lat-compression
!
```

Transparent Bridging Configuration Examples

This section provides example configurations that you can use to configure your bridging environment.

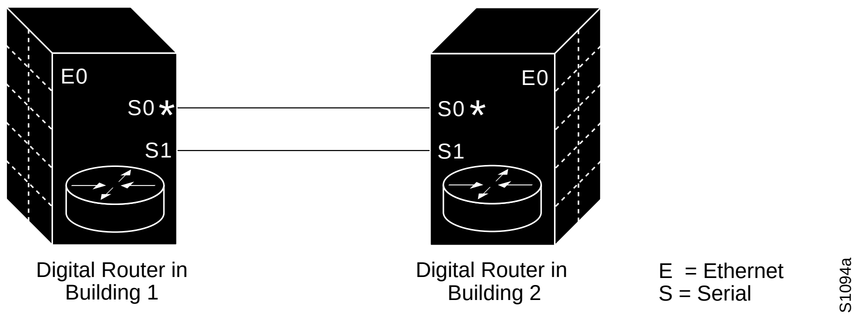
Configuring Ethernet Bridging

In this example, two buildings have networks that must be connected via a T1 link. For the most part, the systems in each building use either IP or DECnet and, therefore, should be routed. There are some systems in each building that must communicate, but they can use only a proprietary protocol.

The example places two Ethernets in each building. One of the Ethernets is attached to the hosts that use a proprietary protocol, and the other is used to attach to the rest of the building network running IP and DECnet. The Ethernet attached to the hosts using a proprietary protocol is enabled for bridging to the serial line and to the other building.

Figure 7–7 shows an example configuration. The interfaces marked with an asterisk (*****) are configured as part of spanning tree 1. The routers are configured to route IP and DECnet. This configuration permits hosts on any Ethernet to communicate with hosts on any other Ethernet using IP or DECnet. In addition, hosts on Ethernet 0 in either building can communicate using protocols not supported for routing.

Figure 7–7 Ethernet Bridging Configuration Example



The configuration file for the router/bridge in Building 1 would be as follows. Note that no bridging takes place over Serial 1. Both IP and DECnet routing are enabled on all interfaces.

```
!
decnet address 3.34
interface ethernet 0
ip address 128.88.1.6 255.255.255.0
decnet cost 10
!
interface serial 0
ip address 128.88.2.1 255.255.255.0
bridge-group 1
decnet cost 10
!
interface serial 1
ip address 128.88.3.1 255.255.255.0
bridge-group 1
decnet cost 10
!
bridge 1 protocol dec
```

The configuration file for the router/bridge in Building 2 is similar.

```
!
decnet address 3.56
!
interface ethernet 0
ip address 128.88.11.9 255.255.255.0
decnet cost 10
!
interface serial 0
ip address 128.88.2.2 255.255.255.0
bridge-group 1
decnet cost 10
!
interface serial 1
ip address 128.88.16.8 255.255.255.0
bridge-group 1
decnet cost 10
!
bridge 1 protocol dec
```

Maintaining the Transparent Bridge

Use the CLEAR BRIDGE command to remove any learned entries from the forwarding database and to clear the transmit and receive counts for any statically configured forwarding entries. The command has this syntax:

clear bridge *group*

The argument *group* is the number you chose to specify a particular spanning tree.

Monitoring the Transparent Bridge

This section describes the EXEC commands you use to obtain displays of activity on the bridged network.

Viewing Entries in the Forwarding Database

Use the SHOW BRIDGE command to view classes of entries in the bridge forwarding database. The command syntax follows.

show bridge [*group*] [*interface*]
show bridge [*group*] [*address* [*mask*]]

The optional argument *group* is the number you chose that specifies a particular spanning tree.

The optional argument *interface* is a specific interface, such as Ethernet 0.

The optional argument *address* is a 48-bit canonical (Ethernet ordered) MAC address. This address can be entered with an optional mask of bits to be ignored, which is specified with the *mask* argument.

Example

In the sample display that follows, the first command would display all entries for hosts reachable via interface Ethernet 0, the second command would display all entries with the vendor code of 0000.0c00.0000, and the third command displays the entry for address 0000.0c00.0e1a. In the fourth command, all entries in the forwarding database would be displayed. In all four examples, the bridge group number has been omitted.

```
show bridge ethernet 0
show bridge 0000.0c00.0000 0000.00FF.FFFF
show bridge 0000.0c00.0e1a
show bridge
```

The following is sample output of the show bridge command:

```
Total of 300 station blocks, 295 free
BG Hash  Address        Action Interface Age RX count TX count
 1 00/0  FFFF.FFFF.FFFF discard   -       P     0       0
 1 09/0  0000.0C00.0009 forward Ethernet0  0     2       0
 1 49/0  0000.0C00.4009 forward Ethernet0  0     1       0
 1 CA/0  AA00.0400.06CC forward Ethernet0  0    25       0
```

The first line of the SHOW BRIDGE output lists the total number of forwarding database elements in the system and the number in the free list. The total number of forwarding elements is expanded dynamically, as needed. The memory to hold bridge entries is allocated in blocks of memory sufficient to hold 300 individual entries. When the number of free entries falls below 25, another block of memory sufficient to hold another 300 entries is allocated. Therefore, the size of the bridge forwarding database is limited to the amount of free memory in the router. Other field descriptions follow in Table 7–2.

Table 7–2 Forwarding Database Display Field Descriptions

Field	Description
BG	The bridging group to which the address belongs.
Address	The canonical (Ethernet ordered) MAC address.
Action	The action to be taken when that address is looked up; choices are to discard or forward the datagram.
Interface	The interface, if any, on which that address was seen.
Age	The number of minutes since a frame was received from or sent to that address. The letter "P" indicates a permanent entry. The letter "S" indicates the system as recorded by the router. On the modular systems, this is typically the broadcast address and the router's own hardware address; on the IGS, this field also will include certain multicast addresses.
RX count	The number of frames received from that address.
TX count	The number of frames sent to that address.

Displaying the Known Spanning-Tree Topology

Use the SHOW SPAN command to display the spanning-tree topology known to the router/bridge. The display includes whether or not LAT group code filtering is in effect. The command has this syntax:

show span

The following is a sample output of the SHOW SPAN command. The first part of the display lists global spanning-tree parameters, followed by port-specific parameters.

```
Bridge Group 1 is executing the DEC compatible spanning tree protocol
 Bridge Identifier has priority 127, address 0000.0c00.4369
 Configured hello time 1, max age 15, forward delay 30
 We are the root of the spanning tree
 Acquisition of new addresses is enabled
 Forwarding of multicast source addresses is disabled
 LAT service filtering is disabled
 Topology change flag not set, detected flag not set
 Times: hold 1, topology change 30, notification 30
     hello 1, max age 15, forward delay 30
 Timers: hello 1, topology change 0, notification 0
 --More--
Port 5 (Ethernet0) of bridge group 1 is forwarding. Path cost 10, priority 0
  Designated root has priority 127, address 0000.0c00.4369
  Designated bridge has priority 127, address 0000.0c00.4369
  Designated port is 5, path cost 0
  Timers: message age 0, forward delay 0, hold 1
  LAT compression is not set
  Input LAT service deny group code list is not set
  Input LAT service permit group code list is not set
  Output LAT service deny group code list is not set
  Output LAT service permit group code list is not set
  Access list for input filtering on type is 201; for LSAP is not set
  Access list for input address filter is not set
  Access list for output filtering on type is not set; for LSAP is not set
  Access list for output address filter is not set
 --More--
Port 6 (Serial4) of bridge group 1 is forwarding. Path cost 64, priority 0
  Designated root has priority 127, address 0000.0c00.4369
  Designated bridge has priority 127, address 0000.0c00.4369
  Designated port is 6, path cost 0
  Timers: message age 0, forward delay 0, hold 1
  LAT compression is set
  Input LAT service deny group code list is not set
  Input LAT service permit group code list is not set
  Output LAT service deny group code list is not set
  Output LAT service permit group code list is not set
  Access list for input filtering on type is 202; for LSAP is not set
  Access list for input address filter is not set
  Access list for output filtering on type is 202; for LSAP is not set
  Access list for output address filter is not set
 --More--
Port 3 (Serial1) of bridge group 1 is down. Path cost 10, priority 0
  Designated root has priority 127, address 0000.0c00.4369
  Designated bridge has priority 127, address 0000.0c00.4369
  Designated port is 7, path cost 0
  Timers: message age 0, forward delay 0, hold 1
  LAT compression is not set
  Input LAT service deny group code list is not set
  Input LAT service permit group code list is not set
  Output LAT service deny group code list is not set
  Output LAT service permit group code list is not set
  Access list for input filtering on type is 201; for LSAP is not set
  Access list for input address filter is not set
  Access list for output filtering on type is not set; for LSAP is not set
  Access list for output address filter is not set
```

Debugging the Transparent Bridge

This section describes the EXEC debugging commands that you use to debug the transparent bridge. For each DEBUG command, there is a corresponding UNDEBUG command to disable the reports.

debug span

Use the DEBUG SPAN command to track changes in the spanning-tree topology. This command is useful for verifying correct operation of the spanning-tree protocol.

debug lat

Use the DEBUG LAT command on a bridge to show group-code filtering actions.

debug lat-packet

Use the DEBUG LAT-PACKET command to list all LAT service advertisements that were forwarded.

Transparent Bridging Global Configuration Command Summary

This section provides a summary of the transparent bridging-specific global configuration commands.

[**no**] **access-list** *list* {**permit** | **deny**} *address-mask*

Prepares access control information for filtering of frames by canonical (Ethernet ordered) MAC address. The argument list is an integer from 700 to 799 selected for the list. The argument *address* and *mask* are 48-bit canonical MAC addresses written in dotted triplet form. The ones bits in the *mask* argument are the bits to be ignored in *address*. The **no** version of the command removes a single access list entry.

[**no**] **access-list** *list* {**permit** | **deny**} *type-code wild-mask*

Prepares access control information for filtering of frames by protocol type. The argument *list* is a user-selectable number between 200 and 299 that identifies the list. The keyword **permit** permits the frame; the keyword **deny** denies the frame. The argument *type-code* is a 16-bit hexadecimal number written with a leading "0x." The argument *wild-mask* is another 16-bit hexadecimal number whose ones bits correspond to bits in the *type-code* argument that should be ignored when making a comparison. The **no** version of the command removes a single access list entry.

[**no**] **bridge** *group* **acquire**

The positive form of this command enables the dynamic learning process and is the default. The argument *group* is the spanning-tree group number.

[**no**] **bridge** *group* **address** *MAC-address* [**forward** | **discard**] [*interface*]

Adds or removes an address from the forwarding database. The argument group is the spanning-tree group number. The argument *MAC-address* is a 48-bit dotted triplet canonical (Ethernet ordered) hardware address such as those displayed by the EXEC SHOW ARP command. The argument *MAC-address* is either a station address, the broadcast address, or a multicast destination address. The optional **forward** keyword enables forwarding of a frame sent from or destined to the specified address. The optional **discard** keyword causes frames sent from or destined to the specified address to be discarded without further processing. The optional argument *interface* specifies an interface after the **discard** or **forward** keyword to indicate the interface on which that address can be reached. Use the **no** version of the command followed by the MAC address to remove an address from the forwarding database.

[**no**] **bridge** *group* **domain** *domain-number*

Enables/disables multiple domain spanning trees. Any number of router /bridges can be placed within the domain. The devices in the domain, and only those devices, will then share spanning-tree information. The argument *group* is the bridge group number and must be a number between one and ten, as specified in the BRIDGE *group* PROTOCOL IEEE command. The keyword domain is required; the argument *domain-number* is a domain number you choose. The default domain number is zero; this is the required domain number when communicating to IEEE bridges that do not support this domain extension. The **no** version of this command chooses the default domain of zero.

Note

This command works only when the bridge group is running the IEEE spanning- tree protocol. Non-Digital bridges may not work correctly on DECbrouter 90 networks containing DECbrouter 90 bridges with nonzero domain numbers.

bridge *group* **forward-time** *seconds*

Sets or returns to the default the forward delay interval or the amount of time spent listening for topology change information after an interface has been activated for bridging and before forwarding actually begins. The argument *group* is the group number assigned to the spanning tree. The argument *seconds* is any value between 10 and 200 seconds. The default value is 30 seconds.

bridge *group* **hello-time** *seconds*

Specifies or returns to the default the interval between Hello Bridge Protocol Data Units (BPDUs). The argument *group* is the group number assigned to the spanning tree. The argument *seconds* is any value between one and ten seconds. The default value is one second.

[**no**] **bridge** *group* **lat-service-filtering**

Enables or disables LAT service filtering. The argument *group* specifies the bridge group. The **no** version of the command disables the use of LAT service filtering on the bridge group.

bridge *group* **max-age** *seconds*

Specifies or removes the interval in which the spanning-tree topology is recomputed when a bridge does not hear BPDUs from the root bridge. The argument *group* is the group number assigned to the spanning tree. The argument *seconds* is the interval the bridge will wait to hear BPDUs from the root bridge. The default interval is 15 seconds.

[**no**] **bridge** *group* **multicast-source**

Allows or disallows the forwarding, but not the learning, of frames with multicast source addresses. The argument *group* is the group number assigned to the spanning tree.

bridge *group* **priority** *number*

Sets the priority of an individual bridge for selection as the root bridge. The argument *group* is the group number assigned to the spanning tree. The argument *number* can range from 1 to 65000. The default priority value is 128. A lower number increases the likelihood for selection as the root bridge.

[**no**] **bridge** *group* **protocol** {**ieee** | **dec**}

Defines or removes a spanning-tree protocol and spanning-tree group. The argument *group* is a number between one and nine that refers to a particular spanning tree. The keyword **protocol** specifies the protocol to use, either **ieee** or **dec**.

Transparent Bridging Interface Subcommand Summary

This section provides an alphabetical summary of the bridging-specific interface subcommands.

[**no**] **bridge-group** *group*

Assigns or removes the network interface to or from the spanning-tree group. The argument *group* is the group number assigned to the spanning tree.

bridge-group *group* **circuit** *number*

Establishes or removes load balancing. The command marks a serial interface as belonging to circuit group number. The argument *group* is the spanning-tree group number.

The argument *number* defines the circuit group number and is a small integer less than ten. Parallel serial interfaces on both bridges must all be flagged as being members of the same circuit group.

[**no**] **bridge-group** *group* **input-address-list** *list*

Assigns or removes an access list to a particular interface for filtering by the MAC source addresses. The argument *group* is the spanning-tree group number. The argument *list* is an access list number between 200 and 299 that you assigned with the bridge ACCESS-LIST command.

bridge-group *group* **input-lat-service-deny** *grouplist*

When enabled, causes the system to not bridge any LAT service advertisement that match the group list specified on input. The argument *grouplist* lists the LAT groups. The argument *group* is the spanning-tree group number. Default is no filtering.

bridge-group *group* **input-lat-service-permit** *grouplist*

When enabled, causes the system to bridge only those LAT service advertisements that match the group list specified on input. The argument *grouplist* lists the LAT group codes. The argument *group* is the spanning-tree group number. Default is no filtering.

bridge-group *group* **input-lsap-list** *list*

Adds or removes a filter for IEEE 802-encapsulated packets on input. This access list is applied to all IEEE 802 frames received on that interface prior to the bridge-learning process. The argument *group* is the spanning-tree group number. The argument *list* is the access list number between 200 and 299 that you assigned with the bridge **access-list** command.

bridge-group *group* **input-type-list** *list*

Adds or removes a filter for Ethernet- and SNAP-encapsulated packets on input. The access list is then applied to all Ethernet frames received on that interface prior to the bridge learning process. The argument *group* is the spanning-tree group number. The argument *list* is the access list number between 200 and 299 that you assigned with the bridge ACCESS-LIST command.

[**no**] **bridge-group** *group* **lat-compression**

Reduces the amount of bandwidth that LAT traffic consumes on serial interfaces. The argument *group* is the spanning-tree group number. Compression is applied to LAT frames being sent out the router/bridge through the interface in question. LAT compression can be specified only for serial interfaces. For the most common LAT operations (user keystrokes and acknowledgment packets), LAT compression reduces LAT's bandwidth requirements by nearly a factor of two.

[**no**] **bridge-group** *group* **output-address-list** *list*

Assigns or removes an access list to a particular interface for filtering by the MAC destination addresses. The argument *group* is the spanning-tree group number. The argument *list* is an access list number between 200 and 299 that you assigned with the bridge ACCESS-LIST command.

[**no**] **bridge-group** *group* **output-lat-service-deny** *grouplist*

When enabled, causes the system to not bridge onto this output interface any LAT service advertisements that match any group in the argument *grouplist*. The argument *group* is the spanning-tree group number. Default is no filtering.

[**no**] **bridge-group** *group* **output-lat-service-permit** *grouplist*

When enabled, causes the system to bridge onto this output interface only those service advertisements that match any group in the argument *grouplist*. The argument *group* is the spanning-tree group number. If a message matches both a deny and a permit, the message will not be bridged. The EXEC SHOW SPAN command reports the group code filtering in effect. Default is no filtering.

bridge-group *group* **output-lsap-list** *list*

Adds or removes a filter for IEEE 802-encapsulated packets on output. This access list is applied just before sending out a frame to an interface. The argument *group* is the spanning-tree group number. The argument *list* is the access list number between 200 and 299 that you assigned with the bridge ACCESS-LIST command.

bridge-group *group* **output-type-list** *list*

Adds or removes a filter for Ethernet- and SNAP-encapsulated packets on output. This access list is then applied just before sending out a frame to an interface. The argument *group* is the spanning-tree group number. The argument *list* is the access list number between 200 and 299 that you assigned with the bridge ACCESS-LIST command.

[**no**] **bridge-group** *group* **path-cost** *cost*

Sets or removes a different path cost. The path cost can range from 0 to 65535, with higher values indicating higher costs. The argument *group* is the spanning-tree group number. The argument *cost* is the path cost. The **no** version of the command chooses the default path cost for the interface.

bridge-group *group* **priority** *number*

Assigns a priority to an interface. This priority is used in tie-breaking when computing a network topology. The argument *group* is the spanning-tree group number. The argument *number* can range from 0 to 255. The default value is zero; the lower the number, the more likely it is that the bridge on this interface will be chosen as the root.

8 Configuring Serial Tunneling and SDLC Transport

This chapter describes the DECbrouter 90's Serial Tunneling (STUN) and SDLC Transport implementations. The following topics are included in this chapter:

- Description of the Serial Tunneling and SDLC Transport functions and procedures for configuration
- Configuring the router/bridge to exchange data over HDLC-compliant links
- Configuring Local Acknowledgment of SDLC sessions
- Configuring local LU address prioritization
- Creating a custom serial transport protocol

This chapter also provides STUN configuration examples and describes how to monitor and debug STUN. Command summaries are included at the end of the chapter.

For a description of SNA terminology refer to the following IBM documents: IBM Corporation, *Synchronous Data Link Control-Concepts*, GA27-3073.
IBM Corporation, *Systems Network Architecture-Technical Overview*, GC30-3073
IBM Corporation, *Systems Network Architecture-Session between Logical Units,* GC20-1868

The Serial Tunnel (STUN) and Serial Transport Functions

The Serial Tunnel (STUN) function allows two devices using SDLC- or HDLC-compliant protocols that are normally connected by a direct serial link, to be connected through one or more DECbrouter 90 or Cisco routers. STUN was designed by Cisco Systems.

The SDLC Transport function, which is a variation of STUN, allows sessions that are using SDLC protocols and TCP/IP encapsulation to be locally terminated. SDLC Transport replaces proxy polling, a method used in prior releases to limit traffic transmitted across a network backbone.

By replacing direct serial links with routers, serial frames can be propagated over arbitrary media and topologies to another DECbrouter 90 with a STUN link to an appropriate end point. The intervening network is not restricted to STUN traffic, but rather, is multiprotocol. For example, instead of running parallel backbones for DECnet and SNA/SDLC traffic, this traffic now can be integrated into an enterprise backbone network.

As another example, using the SDLC protocol, STUN allows networks with IBM mainframes and communications controllers to share data using DECbrouter 90 routers and existing network links. As an SDLC function, STUN fully supports the IBM Systems Network Architecture (SNA), and allows IBM Synchronous Data Link Control (SDLC) frames to be transmitted across the network media and and/or shared serial links. Figure 8–1 illustrates a typical network configuration with and without SDLC serial tunneling.

The software encapsulates SDLC frame traffic into IP packets and routes them over any of the IP-supported network media—serial, FDDI, Ethernet, and Token Ring, X.25, SMDS, and T1/T3—using TCP/IP encapsulation. Because TCP/IP encapsulation is used, you can use any of the DECbrouter 90 routing protocols to route the packets.

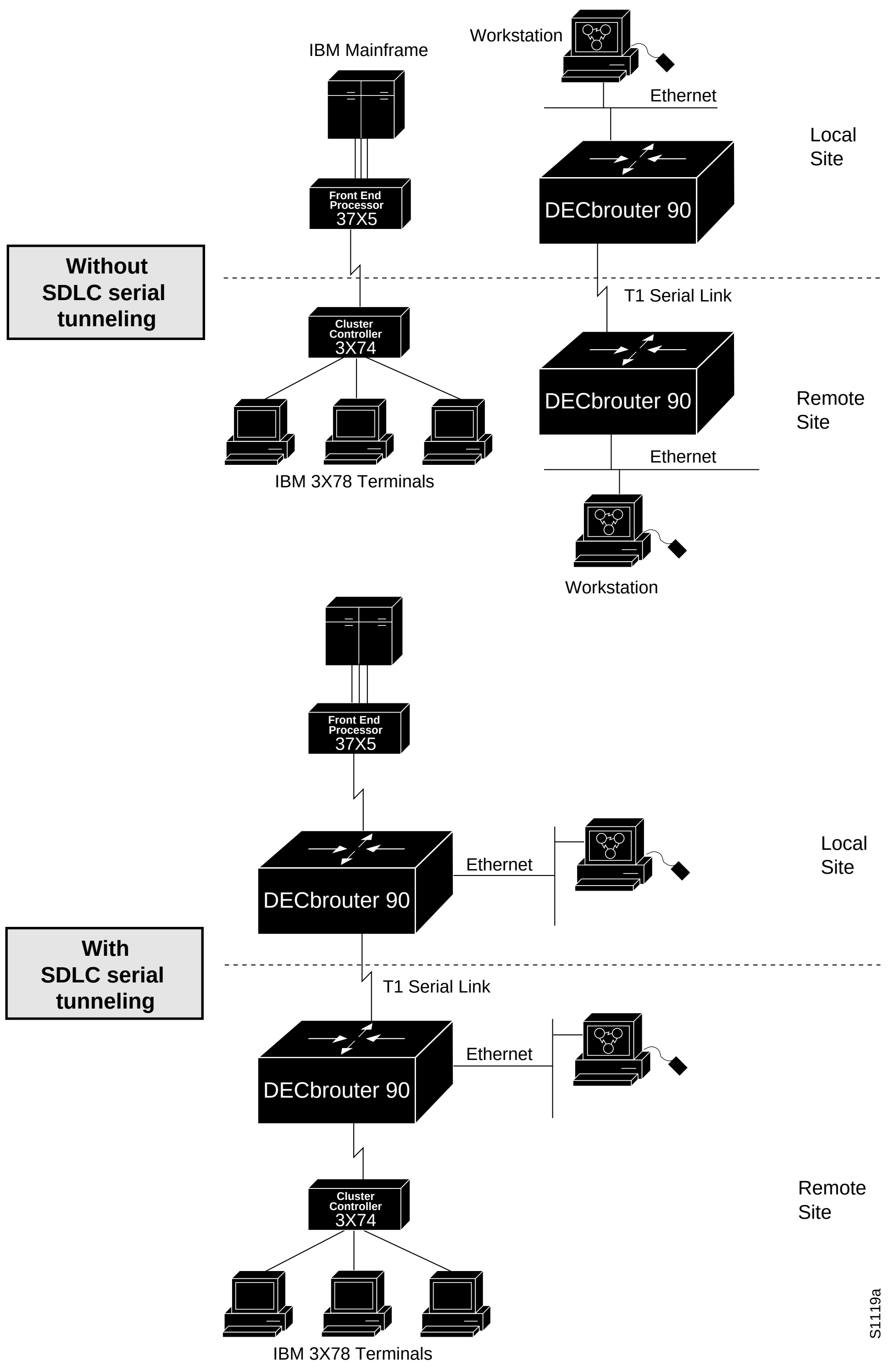
STUN copies frames to destinations based on address, but does not modify the frames in any way or participate in SDLC windowing or retransmission; these functions are left to the communicating hosts. However, SDLC Transport does participate in SDLC windowing and retransmission through local termination of the SDLC session.

SDLC was designed to ensure reliable data transmission across serial media having minimal or predictable time delays. With the advent of STUN and wide area network (WAN) backbones, serial links now can be separated by wide, geographic distances spanning countries and continents. As a result, these serial links have time delays that are longer than SDLC allows for bidirectional communication between hosts. The SDLC Transport in router /bridges supporting STUN addresses the problems of unpredictable time delays, multiple retransmissions, or loss of sessions.

The STUN function also provides for configuration of redundant links to provide transport paths in the event part of the network goes down.

Proxy polling is supported for compatability with prior software releases; however, the functions previously implemented in proxy polling now are more fully supported by the SDLC Local Acknowledgment function of STUN with TCP/IP encapsulation.

Figure 8–1 IBM Network Configuration With and Without SDLC Transport



Configuring STUN

The STUN software provides several methods by which devices using ISO 3309-compliant protocols can exchange data via links connected to one or more routers.

SDLC serial tunneling (STUN) transports SDLC frames across arbitrary, intermediate network media unchanged, thereby expecting the host to take care of SDLC windowing and retransmission functions. In this manner SDLC STUN acts like a multidrop or point-to-point serial line. SDLC frames can be encapsulated in either a TCP/IP encapsulation or an HDLC encapsulation.

Non-SDLC STUN transports arbitrary ISO 3309-compliant frames (such as HDLC) across arbitrary, intermediate network media unchanged. Similar to SDLC STUN, these arbitrary frames can be encapsulated using the TCP/IP encapsulation over any IP network media or HDLC serial encapsulation.

When using SDLC STUN, you can decrease traffic that flows across the backbone network and increase session reliability by optionally configuring local termination, using the Local Acknowledgment feature. When Local Acknowledgment is enabled, you also can define priorities based on the local LU address, enabling one session to have precedence over another.

The software also supports configuration of redundant links and provides commands to monitor and debug the STUN.

The following sections outline the steps you must take to configure your router for these features. These are followed by sections that describe the tasks and provide configuration examples, and the commands to maintain the links. An alphabetically arranged summary of the configuration commands also is provided at the end of the chapter.

Configuring SDLC Serial Tunneling

Follow these steps to configure the SDLC serial tunneling function:

1. Enable STUN and define the transport peers using the STUN PEER-NAME command.
2. Specify the SDLC transport protocol using the STUN PROTOCOL-GROUP command and the sdlc keyword.

The SDLC transport mechanism uses TCP/IP encapsulation and simple serial transport mechanisms. You assign transport peers using IP addresses or serial interface names, and transport groups by assigning group numbers and listing the protocol. Continue with these steps to configure STUN's SDLC transport on the interface:

3. Configure STUN encapsulation on the interface using the ENCAPSULATION STUN command.
4. Specify the group in which the interface will participate using the STUN GROUP command. This step and step 3 together assign the STUN protocol to be used.
5. Define how the frames will be forwarded using the STUN ROUTE command. An option to this command provides local termination using the Local Acknowledgment function.
6. Configure transport-specific features such as local LU address prioritization or SDLC address prioritization, if needed.

Configuring Non-SDLC Serial Tunneling

Follow these steps to configure serial tunneling:

1. Enable STUN and define the transport peers using the STUN PEER-NAME command.
2. Define the protocol to be used. You can choose from predefined protocols, or define your own protocol. (If you define your own protocol, this must be done before step 1 using the STUN SCHEMA command.)
3. Assign the predefined or new protocols to a group using the STUN PROTOCOL-GROUP command.

The non-SDLC transport mechanism uses TCP/IP encapsulation and simple serial transport mechanisms. You assign transport peers using IP addresses or serial interface names, and transport groups by assigning group numbers and listing the protocol. Continue with these steps to configure STUN on the interface:

4. Configure STUN encapsulation on the interface using the ENCAPSULATION STUN command.
5. Specify the group in which the interface will participate using the stun group command. This step and step 4 together assign the STUN protocol to be used.
6. Define how the frames will be forwarded using the STUN ROUTE command.

Notes and Tips About Configuring STUN

Keep the following caveats in mind when configuring STUN:

- The STUN function only will work on full-duplex lines (RTS/CTS always high).
- If you are using the SDLC serial tunneling in a multipoint (multidrop) configuration, you may need to ensure (with cabling) that the Carrier Detect (Receive Line Signal Detect, or RLSD) pin leading into your Primary SNA device is held low.

Enabling Serial Tunneling

Use the STUN PEER-NAME global configuration command to enable the STUN function. The command syntax follows.

stun peer-name *ip-address*
no stun peer-name *ip-address*

Enter the IP address by which this STUN peer is known to other STUN peers that are using the TCP transport for the argument *ip-address*. (Even if you do not use the TCP transport, you must issue this command to define a peer name.)

Use the NO STUN PEER-NAME command with the appropriate IP address to disable the STUN function.

Example

This command assigns IP address 131.108.254.6 as the STUN peer:

```
!
stun peer-name 131.108.254.6
!
```

Defining the STUN Protocol

Each STUN interface is placed in a group that defines the ISO 3309-compliant framed protocol running on that link. Use the STUN PROTOCOL-GROUP global configuration command to define the group number and protocol. The command has this syntax:

> **stun protocol-group** *group-number protocol-keyword*
> **no stun protocol-group** *group-number protocol-keyword*

The STUN PROTOCOL-GROUP command associates group numbers with protocol names. The *group-number* argument can be any number you select between 1 and 255. There are three predefined STUN protocols: **basic**, SDLC, and SDLC transmission group. The protocols are specified by supplying the keyword **basic, sdlc,** or **sdlc-tg** for the *protocol-keyword* argument. You also can define your own STUN protocol. See the section Defining Your Own STUN Protocols.

Use the NO STUN PROTOCOL-GROUP command with the appropriate group number and protocol to remove an interface from the group.

Note

If you are defining a custom protocol, you must do so before doing this step; see the section Defining Your Own STUN Protocols.

Choosing the SDLC Serial Tunneling Protocol

The SDLC serial tunneling protocol is used for placing the routers in the midst of either point-to-point or multipoint (multidrop) SDLC links. Currently, only full-duplex links are supported. An example of how to set up the SDLC STUN protocol follows.

Example

This example command specifies that group 7 use the SDLC STUN protocol:

```
!
stun protocol-group 7 sdlc
!
```

Note

Selecting this predefined protocol allows use of the optional proxy polling feature.

Choosing the Basic Serial Tunneling Protocol

The basic STUN protocol is unconcerned with details of serial protocol addressing and is used when addressing is unimportant. Use this when your goal with STUN is to replace one or more sets of point-to-point (not multidrop) serial links by using a protocol other than SDLC. An example of how to set up the basic STUN protocol follows.

Example

This

```
!
stun protocol-group 5 basic
!
```

Configuring STUN on the Interface

To enable and configure the STUN function on a particular serial interface, use the interface subcommand. The command has this syntax:

encapsulation stun

This command must be specified. It is not possible to further configure the interface without first specifying this command.

Example

These commands enable interface serial 0 for STUN:

```
!
interface serial0
encapsulation stun
!
```

Placing the Interface in a STUN Group

Each STUN-enabled interface on a router must be placed in a previously defined STUN group. Packets will only travel between STUN-enabled interfaces that are in the same group. Use this interface subcommand to do this:

stun group *group-number*
no stun group *group-number*

The argument *group-number* is a number that you assign and that must be a decimal integer between 1 and 255 (inclusive).

Example

These commands place serial interface 1 in STUN group 1, which is defined to run the SDLC transport:

```
!
stun protocol-group 1 sdlc
!
interface serial 1
encapsulation stun
stun group 1
!
```

Note

Once a given serial link is configured for the STUN function, it is no longer a shared multiprotocol link. All traffic that arrives on the link will be transported to the corresponding peer as determined by the current STUN configuration.

Defining How Frames Will Be Forwarded

To define how frames will be forwarded on the interface, use one of the following variations of the STUN ROUTE interface subcommand:

stun route all tcp *ip-address*
no stun route all tcp *ip-address*

stun route all interface serial *interface-number*
no stun route all interface serial *interface-number*

stun route all interface serial *interface-number* **direct**
no stun route all interface serial *interface-number* **direct**

stun route address *address-number* **tcp** *ip-address* [**local-ack**][**priority**]
no stun route address *address-number* **tcp** *ip-address*

stun route address *address-number* **interface serial** *interface-number*
no stun route address *address-number* **interface serial** *interface-number*

stun route address *address-number* **interface serial** *interface-number* **direct**
no stun route address *address-number* **interface serial** *interface-number* **direct**

Use the command forms with the **all** keyword when *all* STUN traffic received on the input interface will be propagated regardless of what address is contained in the serial frame. (These are the only STUN ROUTE command forms allowed with the basic STUN transport protocol.)

The **tcp** keyword causes TCP/IP encapsulation to be used to propagate frames that match the entry. TCP/IP encapsulation allows movement of serial frames across arbitrary media types and topologies. This is particularly useful for building shared, multiprotocol enterprise network backbones. Enter the address that identifies the remote STUN peer that is connected to the far serial link for the *ip-address* argument.

The **local-ack** keyword specifies that SDLC sessions are locally terminated, as described later. The **priority** keyword may be specified along with local-ack to enable priority queuing for the SDLC frames. The prioritization feature does add overhead, so be selective in its use.

The **interface serial** keywords cause the HDLC encapsulation to be used to propagate the serial frame. There must be an appropriately configured router on the other end of the designated serial line. The outgoing serial link still can be used for other kinds of traffic (the frame is not TCP encapsulated). This mode is used when TCP/IP encapsulation is not needed or when higher performance is required. Enter the serial line number connected to the router for the *interface-number* argument.

Use the command forms with the **address** keyword to specify how a serial frame that contains a particular address is to be propagated. The address you enter for the *address-number* argument varies with the protocol type. The argument will accept octal, decimal, or hexadecimal addresses, as appropriate, in the range allowed by the protocol. As an example, SDLC uses hexadecimal digits and has a one-byte address field. The allowable addresses for this protocol must be between 00 and FF, inclusive.

It is possible to use both the **all** and **address** keywords for the same input serial interface. When this is done, the address specifications take effect first. If none of these match, the **all** keyword will be used to propagate the frame.

Use the command form with the **direct** keyword at the end of the command to indicate that the specified interface is also a direct STUN link, rather than a serial connection to another peer.

STUN Class of Service

The global command STUN COS-ENABLE specifies "Class of Service." The syntax of the command is as follows:

stun cos-enable
no stun cos-enable

Specifying **cos-enable** will force the router to read the contents of the Format Identification 4 (FID4) frames to prioritize traffic. All FEP-to-FEP (Front-End Processor) frames contain priority information. This priority information is encoded in two bits of the FID4 header. The router will read the two bits of the frame if it is an FID4 frame and will decode the priority information. Once decoded, the frames will be placed on the correct priority queue to be sent out on the backbone network. Thus, with this feature, you can prioritize your SNA traffic allowing your important FEP traffic to flow on high-priority queues. Note that this is useful only between FEP-to-FEP (PU4-to-PU4) communications across the non-SNA backbone.

When a PU4 device communicates with a PU2/PU2.1 device, it uses FID2 frames that do not contain COS or priority information. This priority traffic is only for the backbone network that separates the "peers."

Note

Local Acknowledgment for STUN must be turned on and the STUN ROUTE ADDRESS PRIORITY command must be issued for the COS feature to take effect.

Configuring STUN Serial Link Address Priorities

Use the PRIORITY-LIST global configuration command to establish queuing priorities based on the address of the serial link on a STUN connection. The full syntax of this command follows.

priority-list *list* **stun** *queue-keyword address group-number address-number*
no priority-list *list* **stun** *queue-keyword address group-number address-number*

The argument *list* is an arbitrary integer between 1 and 10 that identifies the priority list selected by the user.

The keyword **stun** indicates that this is a serial tunnel feature.

The argument *queue-keyword* is a priority queue name, one of high, medium, normal, or low.

The keyword *address* is used with the two arguments *group-number* and *address-number* that uniquely identify a STUN serial link.

The argument *group-number* is the group number used in the STUN GROUP interface configuration subcommand.

The argument *address-number* is the address of the serial link. The format of the address number is either a one-byte hex value (for example, C1) for a SDLC link or one that is specified by the STUN SCHEMA configuration command.

Configuring Serial Link Address Prioritization on STUN Simple Serial Transport

The PRIORITY-GROUP interface subcommand is used to assign a priority group to the output interface.

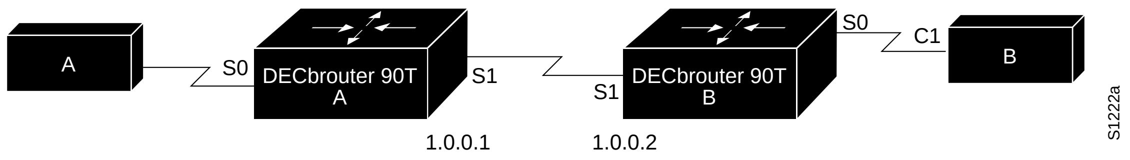
priority-group *list*
no priority-group *list*

The argument *list* is the priority list number of the output interface.

Example

Assume that the link between DECbrouter 90 A and DECbrouter 90 B is a serial tunnel that uses the simple serial transport mechanism as shown in Figure 8–2.

Figure 8–2 STUN Simple Serial Transport



Device A is to communicate with device C (SDLC address C1) with priority high. Device B is to communicate with device D (SDLC address A7) with normal priority. The router configurations would be as follows:

Router A

```
stun peer-name 1.0.0.1
stun protocol-group 1 sdlc
!
interface serial 0
no ip address
encapsulation stun
stun group 1
stun route address C1 interface serial 1
!
interface serial 1
ip address 1.0.0.1 255.0.0.0
priority-group 1
!
priority-list 1 stun high address 1 C1
!
```

Router B

```
stun peer-name 1.0.0.2
stun protocol-group 1 sdlc
!
interface serial 0
no ip address
encapsulation stun
stun group 1
stun route address C1 interface serial 1
!
interface serial 1
ip address 1.0.0.2 255.0.0.0
priority-group 1
!
priority-list 1 stun high address 1 C1
!
```

Configuring Serial Link Address Prioritization on STUN TCP/IP Encapsulation

The PRIORITY-GROUP interface subcommand is used to assign a priority group to the input interface. Rather than using PRIORITY-GROUP as an output interface subcommand, it is used as an *input* interface subcommand for STUN. This is because the IP network could be connected to multiple input interfaces of the router.

priority-group *list*
no priority-group *list*

The argument *list* is the priority list number of the input interface.

You must use the PRIORITY-LIST command, to assign priorities to the ports as shown in Table 8–1.

Table 8–1 STUN Priority Port Numbers

Priority	Port
STUN high priority	1994
STUN medium priority	1990
STUN normal priority	1991
STUN low priority	1992

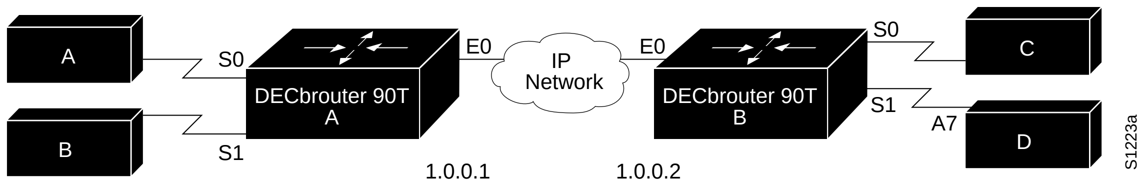
Note

Both Local Acknowledgment and TCP priority features for STUN must be turned on in order for serial link address prioritization to take effect.

Example

Assume the link between Digital A and Digital B is a serial tunnel that uses the TCP/IP encapsulation as shown in Figure 8–3.

Figure 8–3 STUN TCP/IP Encapsulation



Device A is to communicate with device C (SDLC address C1) with priority high. Device B is to communicate with device D (SDLC address A7) with normal priority. The router configuration would be as follows:

Router A

```
stun peer-name 1.0.0.1
stun protocol-group 1 sdlc
stun protocol-group 2 sdlc
!
interface serial 0
no ip address
encapsulation stun
stun group 1
stun route address C1 tcp 1.0.0.2 local-ack priority
priority-group 1
!
interface serial 1
no ip address
encapsulation stun
stun group 2
stun route address A7 tcp 1.0.0.2 local-ack priority
priority-group 2
!
interface ethernet 0
ip address 1.0.0.1 255.0.0.0
!
priority-list 1 protocol ip high tcp 1994
priority-list 1 protocol ip medium tcp 1990
priority-list 1 protocol ip normal tcp 1991
priority-list 1 protocol ip low tcp 1992
priority-list 1 stun low address 1 C1
!
priority-list 2 protocol ip high tcp 1994
priority-list 2 protocol ip medium tcp 1990
priority-list 2 protocol ip normal tcp 1991
priority-list 2 protocol ip low tcp 1992
priority-list 2 stun low address 2 A7
!
hostname DECbrouter90A
router igrp
network 1.0.0.0
```

Router B

```
stun peer-name 1.0.0.2
stun protocol-group 1 sdlc
stun protocol-group 2 sdlc
!
interface serial 0
no ip address
encapsulation stun
stun group 1
stun route address C1 tcp 1.0.0.1 local-ack priority
priority-group 1
!
interface serial 2
no ip address
encapsulation stun
stun group 2
stun route address A7 tcp 1.0.0.1 local-ack priority
priority-group 2
!
interface ethernet 0
ip address 1.0.0.2 255.0.0.0
!
priority-list 1 protocol ip high tcp 1994
priority-list 1 protocol ip medium tcp 1990
priority-list 1 protocol ip normal tcp 1991
priority-list 1 protocol ip low tcp 1992
priority-list 1 stun low address 1 C1
!
priority-list 2 protocol ip high tcp 1994
priority-list 2 protocol ip medium tcp 1990
priority-list 2 protocol ip normal tcp 1991
priority-list 2 protocol ip low tcp 1992
priority-list 2 stun low address 2 A7
!
hostname DECbrouter90B
router igrp 109
network 1.0.0.0
```

SDLC Local Acknowledgment

The Serial Tunneling function and encapsulation protocol selection are described earlier in this chapter. SDLC STUN with TCP/IP encapsulation must be enabled in order to use SDLC Local Acknowledgment.

SDLC connections for STUN can be terminated locally at the router. By locally acknowledging these sessions, you can avoid the delays of the wide area network that can affect normal STUN or STUN with proxy polling. SDLC Local Acknowledgment removes all of the SDLC session keepalives (nonproductive Receiver Ready (RR) exchanges) from the wide area network, resulting in reduced overhead on the IP network.

With SDLC Local Acknowledgment, also known as local termination, the end-to-end connection is composed of three sessions:

- The primary SDLC session between the SNA device and the router
- The TCP/IP session across the internet
- The secondary SDLC session between the SNA device and the router

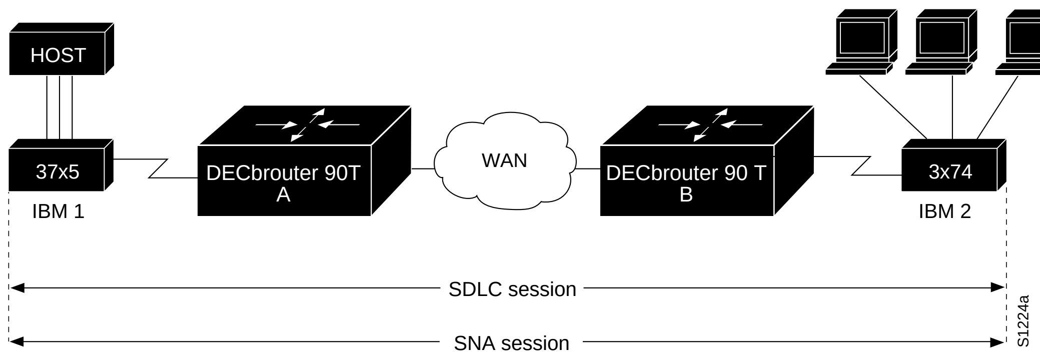
These three sessions are concatenated to form a logical end-to-end connection.

Although this feature is part of the STUN feature set, the SDLC Local Acknowledgment capability applies only to SDLC tunneling across TCP/IP networks. The current release of SDLC Local Acknowledgment does not apply to HDLC tunneling or to transfer of SDLC across HDLC serial links. Functionally, it is the SDLC counterpart of the LLC2 Local Acknowledgment feature for Token Ring networks.

The default mode for STUN is straight passthrough of all SDLC connection traffic (including RRs) end-to-end between SNA end devices. To invoke SDLC local termination, the **local-ack** parameter must be specified on the STUN command that defines the tunnel.

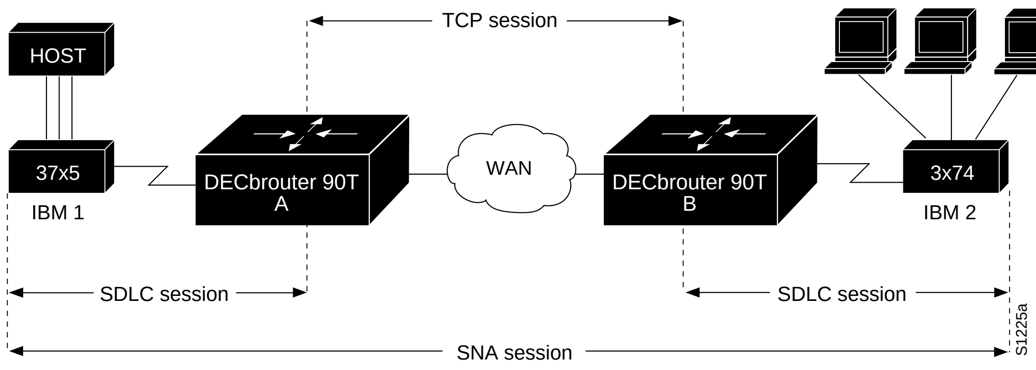
Figure 8–4 illustrates an SDLC session. IBM 1 (for example, a 37x5) using a serial link can communicate with IBM 2 (for example, a 3x74) on a different serial link separated via a wide area backbone network. Frames are transported between Router A and Router B using STUN. However, the SDLC session between IBM 1 and IBM 2 is still end-to-end; that is, every frame generated by IBM 1 traverses the backbone network to IBM 2, and IBM 2, on receipt of the frame, acknowledges it.

Figure 8–4 SDLC Session Without Local Acknowledgment



With Local Acknowledgment turned on, the SDLC session between the two IBM end nodes is not end-to-end but instead terminates at the two local routers, as shown in Figure 8–5. The SDLC session with IBM 1 ends at Router A, and the SDLC session with IBM 2 ends at Router B. Both Router A and Router B execute the full SDLC protocol as part of SDLC Local Acknowledgment.

Figure 8–5 SDLC Session With Local Acknowledgment



With SDLC Local Acknowledgment enabled in both routers, Router A acknowledges frames received from IBM 1. The node IBM 1 treats the acknowledgments it receives as if they are from IBM 2. Similarly, Router B acknowledges frames received from IBM 2.

The node IBM 2 treats the acknowledgments it receives as if they are from IBM 1. With local acknowledgment, frames no longer travel the WAN backbone networks to be acknowledged. This means the end machines do not time out, resulting in no loss of sessions.

The advantages of enabling SDLC Local Acknowledgment are as follows:

- SDLC Local Acknowledgment solves the session keepalive timer problem without having to change any configuration on the end nodes. Local acknowledgment of the sessions is invisible to the end nodes. The primary benefit of this local acknowledgment is that the host's resource definition (such as the sysgen) does not have to change. All the frames acknowledged by the router appear to the end hosts to be coming from the remote IBM machine. In fact, by looking at a trace from a protocol analyzer, it is impossible to tell whether a frame was acknowledged by the local router or by a remote IBM machine.
- All the supervisory frames (RR, RNR, REJ) frames that are locally acknowledged go no farther than the router.

Without Local Acknowledgment, *every* frame traverses the backbone. With SDLC Local Acknowledgment, some control frames are exchanged between remote peers at session initiation, but only data (I-frames) traverse the backbone, resulting in less traffic on the backbone network. For installations in which customers pay for the amount of traffic passing through the backbone, this could be a definite cost-saving measure. A simple protocol exists between the two peers to bring a TCP session up or down.

Configuring Local Acknowledgment for STUN Interfaces

The default mode for STUN is straight passthrough of all SDLC traffic (including RRs) end-to-end between SNA devices. In order to activate SDLC Local Acknowledgment, the **local-ack** parameter of the STUN ROUTE ADDRESS command must be specified on the STUN command that defines the tunnel.

Local Acknowledgment can only be enabled with TCP remote peers (as opposed to LAN or direct serial-interface remote peers), because the routers need the reliable transmission of TCP to provide the same reliability of the pre-Local Acknowledgment end-to-end connection. Therefore, the direct media encapsulation options for the STUN ROUTE ADDRESS command cannot be used.

If the SDLC session between the local host and the router terminates in either router, the other router will be informed to terminate its connection to its local host.

If the TCP queue length of the connection between the two routers reaches 90 percent of its limit, the routers will send Receiver-Not-Ready (RNR) messages to the local hosts until the queue limit is reduced below this limit.

The configuration of the SDLC parameters for the local serial interfaces can affect overall performance. Refer to Chapter 9 for more details about fine-tuning your network through the SDLC parameters.

Setting Up Local Acknowledgment

To set up SDLC Local Acknowledgment for a serial tunnel, specify the **local-ack** keyword of the interface STUN ROUTE ADDRESS command.

stun route address *address-number* **tcp** *ip-address* [**local-ack**] [*priority*]

Setting Up Local Acknowledgment on a Per-STUN-Peer Basis

The STUN ROUTE ADDRESS subcommand provides the options **local-ack** and **priority**. The keyword local-ack indicates that SDLC sessions destined to a specific STUN peer are to be locally acknowledged. The keyword **priority** allow prioritization features such as SNA class of services, SNA LU address prioritization, and SDLC address prioritization to function on STUN over TCP/IP encapsulation.

The following example shows a sample configuration for a pair of Digital routers performing SDLC Local Acknowledgment:

Router A

```
..
stun peer-name 150.136.64.92
stun protocol-group 1 sdlc
...
interface Serial 0
no ip address
encapsulation stun
stun group 1
stun sdlc-role secondary
sdlc address C1
stun route address C1 tcp 150.136.64.93 local-ack
clockrate 19200
...
```

Router B

```
...
stun peer-name 150.136.64.93
stun protocol-group 1 sdlc
...
interface Serial 0
no ip address
encapsulation stun
stun group 1
stun sdlc-role primary
sdlc address C1
stun route address C1 tcp 150.136.64.92 local-ack
clockrate 19200
...
```

Displaying Local Acknowledgment Statistics

The SHOW STUN command shows the current state of any current Local Acknowledgment connections.

show stun

Example

```
router# show stun

This peer: 150.136.64.93

 *Serial0  (group 1 [sdlc])
                            state      l  rx_pkts   tx_pkts      drops poll
C1     TCP 150.136.64.92    open       *      922       343          0

SDLC Local Acknowledgement:

 *Serial0  (group 1 [sdlc])
                                 slack_state conn disc iframe_s iframe_r
C1     TCP 150.136.64.92              Active    2    1      918      340
dino#
```

Table 8–2 describes the fields shown in the preceding example.

Table 8–2 Show Stun Field Descriptions

Field	Description
slack_state	The current state of this locally terminated SDLC
The normal values for slack_state are:	
Active	The locally terminated SDLC session is currently active and able to pass I-Frames between STUN peers.
Disconnected	The session is currently disconnected.
Other values for the slack_state session are:	
AwaitLinkupRequest	An SNRM has been received and a LinkupRequest has been sent to the STUN peer. This side of the session is awaiting a LinkupResponse.
AwaitSdlcOpen	A LinkupRequest has been received from the STUN peer and a SNRM has been sent to the SDLC device. This side of the session is awaiting a UA response from the SDLC device.
AwaitLinkdownResponse	An RD has been received from the SDLC device and a RequestDisconnectRequest has been sent to the STUN peer. This side is awaiting a LinkdownRequest from the STUN peer.
AwaitLinkdownUa	A LinkdownRequest has been received from the STUN peer and a DISC has been sent to the SDLC device. This side is awaiting a UA from the SDLC device.
AwaitLinkdownResponse	A DISC has been received from the SDLC device and a LinkdownRequest has been sent to the STUN peer. This side is awaiting a LinkdownResponse from the STUN peer.
AwaitDisc	An AbortRequest has been received from the STUN peer and an RD has been sent to the SDLC device. This side is awaiting a DISC from the SDLC device.
AwaitAbortResponse	The SDLC session has been lost and an AbortRequest has been sent to the STUN peer. This side is awaiting an AbortResponse from the STUN peer.
conn	The number of times this SDLC session has gone through the session initiation sequence (the SNRM/UA exchanges).
disc	The number of times this SDLC session has gone through the session termination sequence (the DISC/UA exchanges).
iframe_s	The number of I-Frames that have been received from the STUN peer and sent to the SDLC device.
iframe_r	The number of I-Frames that have been received from the SDLC device and sent to the STUN peer.

Debugging Local Acknowledgment

There is one DEBUG command for Local Acknowledgment: DEBUG SDLC-LOCAL-ACK.

```
debug sdlc-local-ack
```

This command displays Local Acknowledgment debugging information that is useful for Digital staff.

There is a corresponding UNDEBUG command that stops the output. There are three levels of debug output, as shown in Table 8–3. The numbers 1,2,4, and 7 are bit flags that you can combine in any manner. The default is 7 for all events.

Table 8–3 Debug SDLC Local-Ack Debugging Levels

Debug Command	Meaning
debug sdlc-local-ack 1	Only U-Frame events
debug sdlc-local-ack 2	Only I-Frame events
debug sdlc-local-ack 4	Only S-Frame events
debug sdlc-local-ack 7	All SDLC Local Ack events (default setting)

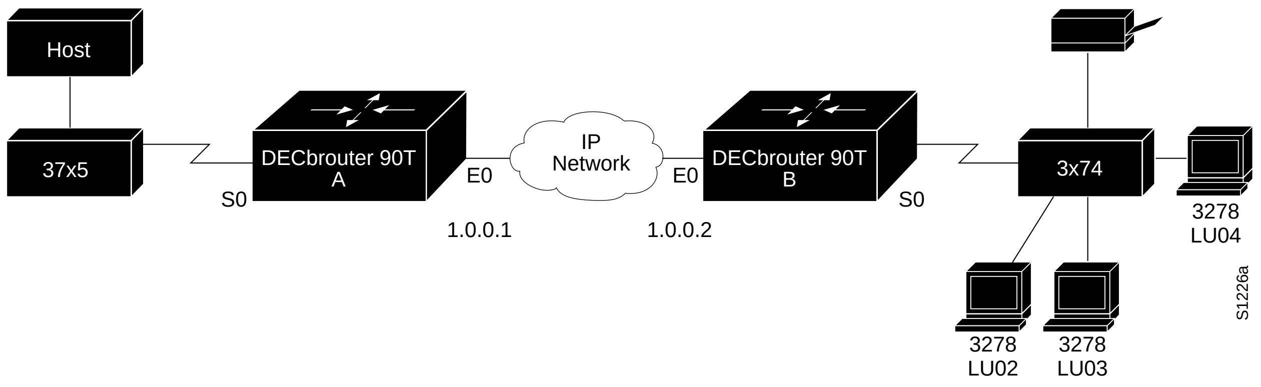
SNA Local LU Address Prioritization

SNA Local LU Address Prioritization is specific to IBM SNA connectivity and is used to prioritize SNA traffic on either Serial Tunnel (STUN) or Remote Source-Route Bridging (RSRB).

The SNA Local LU address prioritization feature allows SNA traffic to be prioritized according to the address of the Logical Units (LU) on the FID2 transmission headers. Currently, only dependent LUs are supported. The prioritization takes place on LU-LU traffic between an SNA Node type 5 or Node type 4, and Node type 2.

An example on the use of SNA Local Address Prioritization is shown in Figure 8–6.

Figure 8–6 SNA Local Address Prioritization



The IBM mainframe is channel-attached to a 3745 communications controller that is connected to a 3174 cluster controller via the serial tunnel. Multiple 3270 terminals and printers, each with a unique local LU address, are then attached to the 3174. By applying SNA Local LU Address Prioritization, each LU associated with a terminal or printer can be assigned a priority; that is, certain users can have terminals that have better response time than others, and printers can have lowest priority.

SNA Local LU Address Prioritization works on both the serial tunnel (STUN) and Remote Source-Route Bridging (RSRB). The configuration commands are described in the next section.

Note

Both Local Acknowledgment and TCP priority features for STUN or RSRB must be turned on for this feature to take effect.

Assigning Priority by SNA Local LU Address

Use the LOCADDR-PRIORITY-LIST global configuration command to establish queuing priorities based on the address of the logical unit. The full syntax of this command follows.

locaddr-priority-list *list address-number queue-keyword*
no locaddr-priority-list *list address-number queue-keyword*

The argument *list* is an arbitrary integer between 1 and 10 that identifies the LU address priority list selected by the user.

The argument *address-number* is the value of the `LOCADDR=` parameter on the LU macro, which is a one-byte address of the LU in hex.

The argument *queue-keyword* is a priority queue name, one of high, medium, normal, or low.

Assigning LOCADDR Priority Groups

The LOCADDR-PRIORITY interface subcommand is used to assign a priority group to an input interface.

locaddr-priority *list*
no locaddr-priority *list*

The argument *list* is the priority list number of the input interface.

You must use the priority-list command, to assign priorities to the ports as shown in Table 8–4.

Table 8–4 STUN Priority Port Numbers

Priority	Port
STUN high priority	1994
STUN medium priority	1990
STUN normal priority	1991
STUN low priority	1992

Example

Example The configuration for the network as illustrated in Figure 23-6 would be as follows:

Router A

```
stun peer-name 1.0.0.1
stun protocol-group 1 sdlc
!
interface serial 0
no ip address
encapsulation stun
stun group 1
stun route address C1 tcp 1.0.0.2 local-ack priority
clockrate 19200
locaddr-priority 1
priority-group 1
!
interface Ethernet 0
ip address 1.0.0.1 255.255.255.0
!
locaddr-priority-list 1 02 high
locaddr-priority-list 1 03 high
locaddr-priority-list 1 04 medium
locaddr-priority-list 1 05 low
!
!
priority-list 1 protocol ip high tcp 1994
priority-list 1 protocol ip medium tcp 1990
priority-list 1 protocol ip normal tcp 1991
priority-list 1 protocol ip low tcp 1992
```

Router B

```
stun peer-name 1.0.0.2
stun protocol-group 1 sdlc
!
interface serial 0
no ip address
encapsulation stun
stun group 1
stun route address C1 tcp 1.0.0.1 local-ack priority
clockrate 19200
locaddr-priority 1
priority-group 1
!
interface Ethernet 0
ip address 1.0.0.2 255.255.255.0
!
locaddr-priority-list 1 02 high
locaddr-priority-list 1 03 high
locaddr-priority-list 1 04 medium
locaddr-priority-list 1 05 low
!
priority-list 1 protocol ip high tcp 1994
priority-list 1 protocol ip medium tcp 1990
priority-list 1 protocol ip normal tcp 1991
priority-list 1 protocol ip low tcp 1992
```

Using STUN with Multiple-Link Transmission Groups

Multilink SDLC Transmission Groups (TG) can be accommodated across STUN connections between IBM communications controllers such as IBM 37x5s.

Serial links that are a part of a Transmission Group are identified to the router with the protocol group **sdlc-tg**. As a result, there is a protocol parameter value for the STUN PROTOCOL-GROUP global command.

stun protocol-group *group-number* **sdlc-tg**

STUN connections that comprise a Transmission Group are merely identified as being members of the appropriate **sdlc-tg** group. Additionally, any STUN connections that are members of an **sdlc-tg** group must be locally acknowledged; that is, **local-ack** must be specified as a parameter on the STUN ROUTE INTERFACE command.

Because SNA Transmission Groups are parallel links to the same pair of IBM communications controllers, all STUN connections that comprise a particular TG must go to the same IP address.

A sample configuration for a pair of routers performing TG support for a two-link Transmission Group follows.

Router A

```
...
stun peer-name 150.136.64.92
stun protocol-group 3 sdlc-tg
...
interface Serial 0
no ip address
encapsulation stun
stun group 3
stun sdlc-role secondary <
sdlc address C1
stun route address C1 tcp 150.136.64.93 local-ack
clockrate 19200
!
interface Serial 1
no ip address
encapsulation stun
stun group 3
stun sdlc-role secondary <
sdlc address C2
stun route address C2 tcp 150.136.64.93 local-ack
clockrate 19200
...
```

Router B

```
...
stun peer-name 150.136.64.93
stun protocol-group 3 sdlc-tg
...
interface Serial 0
no ip address
encapsulation stun
stun group 3
stun sdlc-role primary
sdlc address C1
stun route address C1 tcp 150.136.64.92 local-ack
clockrate 19200
!
interface Serial 1
no ip address
encapsulation stun
stun group 3
stun sdlc-role primary
sdlc address C2
stun route address C2 tcp 150.136.64.92 local-ack
clockrate 19200
...
```

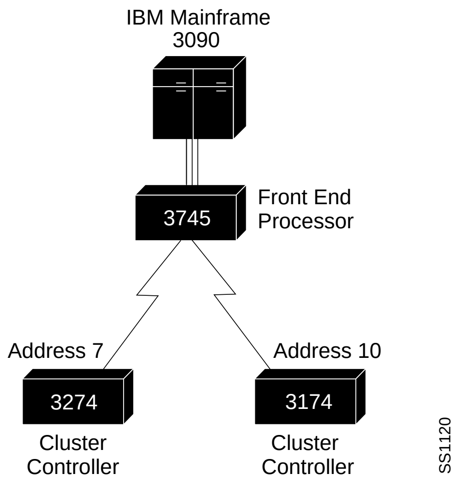
STUN Configuration Examples

This section contains configuration examples illustrating use of the commands described in this chapter.

Expanding the IBM Network Capability Using Digital Routers

Figure 8–7 depicts a typical IBM network configuration.

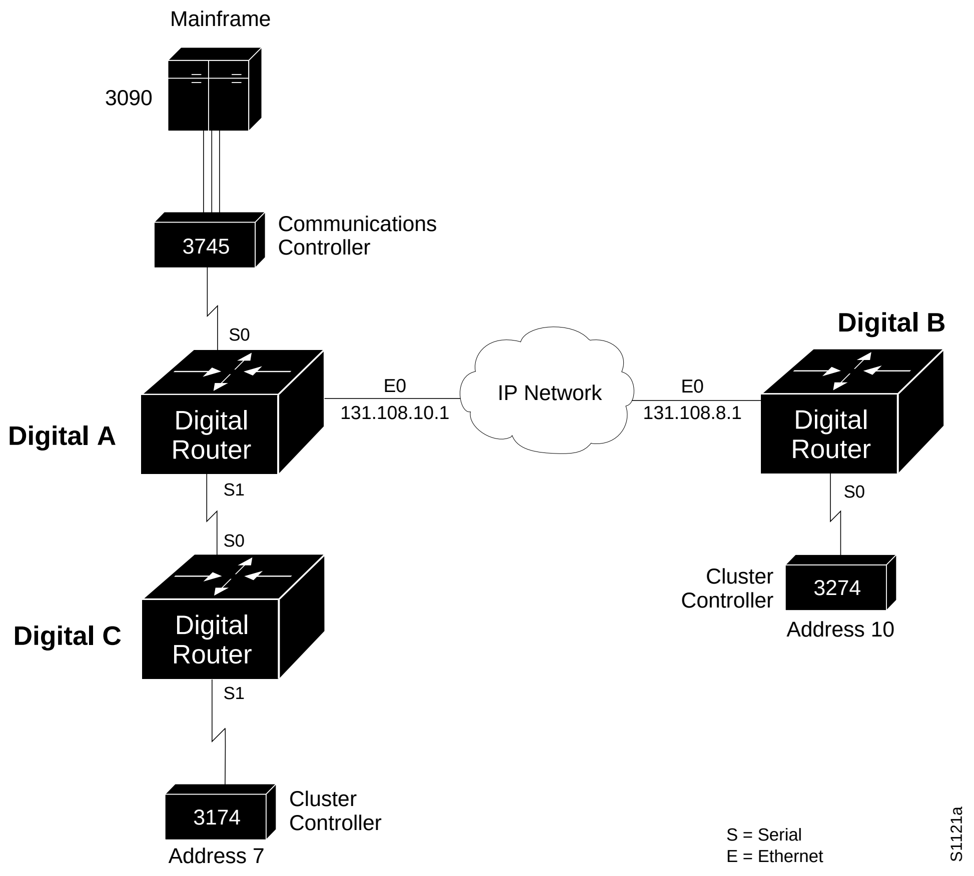
Figure 8–7 IBM Network Without Router



The configuration has a 3090 mainframe channel-attached to a 3745 controller. These are connected by SDLC link to 3274 and 3174 cluster controllers.

Figure 8–8 modifies the configuration by placing the routers between the devices connected via SDLC link, thus expanding the network capabilities. The configuration files for connecting the routers to the IBM network follow.

Figure 8–8 IBM Network With Routers and SDLC Links



Configuration for Router A

```
!
stun peer-name 131.108.10.1
stun protocol-group 1 sdlc
!
interface serial 0
encapsulation stun
no ip address
no keepalive
stun group 1
stun route address 7 interface serial 1
stun route address 10 tcp 131.108.8.1
!
interface serial 1
ip address 131.108.62.1 255.255.255.0
!
interface ethernet 0
ip address 131.108.10.1 255.255.255.0
!
hostname DigitalA
router igrp 161
network 131.108.0.0
```

Configuration for Router B

```
!
stun peer-name 131.108.8.1
stun protocol-group 1 sdlc
!
interface serial 0
encapsulation stun
no ip address
no keepalive
stun group 1
stun route address 10 tcp 131.108.10.1
!
interface ethernet 0
ip address 131.108.8.1 255.255.255.0
!
hostname DigitalB
router igrp 161
network 131.108.0.0
```

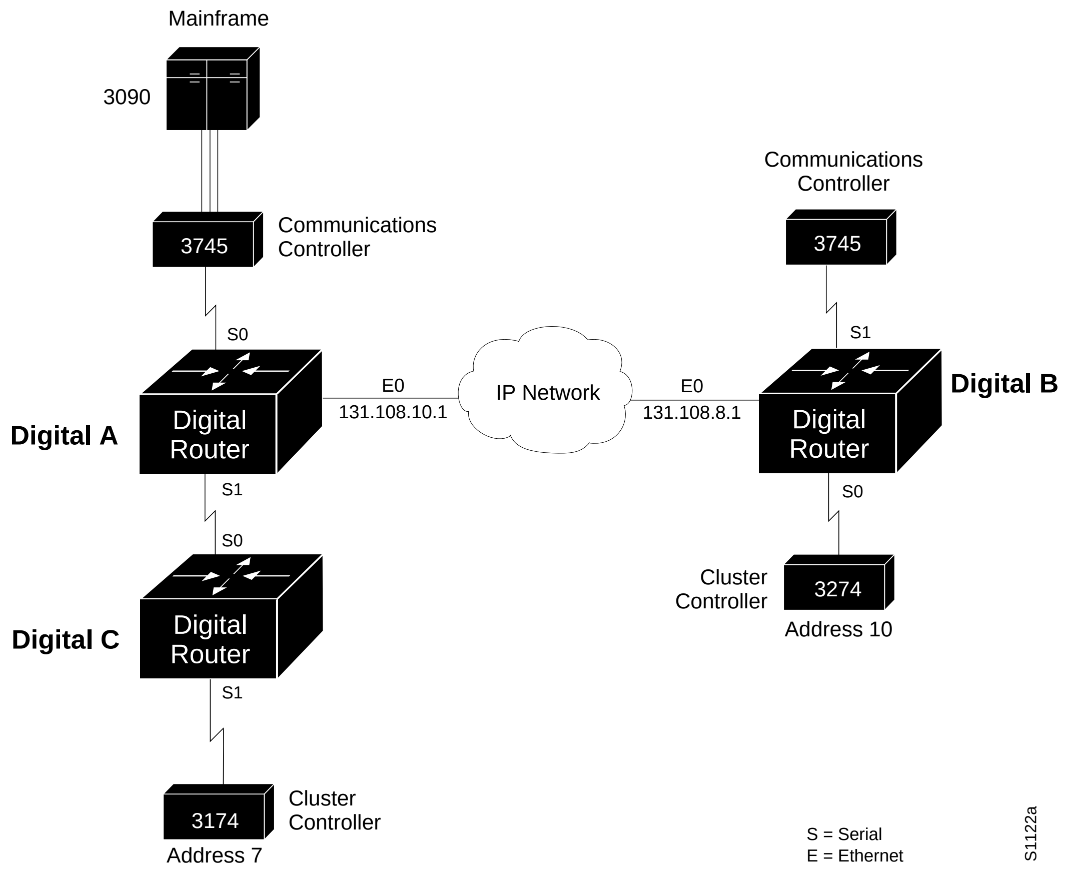
Configuration for Router C

```
!
stun peer-name 131.108.62.2
stun protocol-group 1 sdlc
!
interface serial 0
ip address 131.108.62.2 255.255.255.0
!
interface serial 1
encapsulation stun
no ip address
no keepalive
stun group 1
stun route address 7 interface serial 0
!
hostname DigitalC
router igrp 161
network 131.108.0.0
```

Extended IBM Network

As another example, the previous configuration can be extended by connecting a remote 3745 to another serial interface connected to Router B. All other traffic from the primary node (the 3745 connected to the 3090), other than addresses 7 and 10, should go to this remote 3745. Figure 8–9 illustrates this configuration.

Figure 8–9 Extended IBM Network with Routers and SDLC Links



The following configuration changes would need to be made to the serial interface in the router labeled Router B.

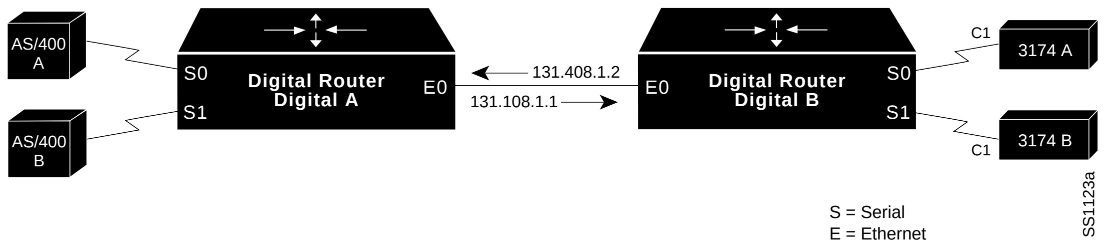
```
!
interface serial 1
encapsulation stun
no ip address
no keepalive
stun group 1
stun route all tcp 131.108.10.1
!
```

Serial 0 on Router A now looks like this:

```
!
interface serial 0
encapsulation stun
no ip address
no keepalive
stun group 1
stun route address 7 interface serial 1
stun route address 10 tcp 131.108.8.1
stun route all tcp 131.108.8.1
!
```

Notice in the preceding examples that the same STUN group number is used in all cases. If these numbers differ between the three routers, attempts at communication will be unsuccessful. There are times, however, when having multiple groups can be useful. Such an example is illustrated in Figure 8–10.

Figure 8–10 IBM Network with Multiple Groups of Controllers



In the following configuration, the AS/400 device labeled A wants to communicate with the 3174 device labeled A, as do the AS/400 device labeled B and 3174 device labeled B. Notice that both of the 3174 devices are at SDLC address C1. A first attempt at the configuration files for the routers labeled A and B could be as follows:

Configuration for Router A

```
!
stun peer-name 131.108.1.1
stun protocol-group 1 sdlc
!
interface ethernet 0
ip address 131.108.1.1 255.255.0.0
!
interface serial 0
encapsulation stun
no ip address
no keepalive
stun group 1
stun route address C1 tcp 131.108.1.2
!
interface serial 1
encapsulation stun
no ip address
no keepalive
stun group 1
stun route address C1 tcp 131.108.1.2
!
```

Configuration for Router B

```
!
stun peer-name 131.108.1.2
stun protocol-group 1 sdlc
!
interface ethernet 0
ip address 131.108.1.2 255.255.0.0
!
interface serial 0
encapsulation stun
no ip address
no keepalive
stun group 1
stun route address C1 tcp 131.108.1.1
!
interface serial 1
encapsulation stun
no ip address
no keepalive
stun group 1
stun route address C1 tcp 131.108.1.1
!
```

A problem occurs when the router labeled A transfers an SDLC frame with address C1 from the AS/400 A device to the router labeled B. There would be no way to determine whether it came from the AS/400 device A and was therefore destined to the 3174 device A, or that it came from the AS/400 device B and was therefore destined to the 3174 device B.

The use of distinct group numbers solves this problem. The next configuration will ensure correct communication, because frames that come in on group 1 links will only go out of group 1 links. The same holds true for frames coming in on group 2, or any other numbered group link. The new configuration follows.

Configuration for Router A

```
!
stun peer-name 131.108.1.1
stun protocol-group 1 sdlc
stun protocol-group 2 sdlc
!
interface ethernet 0
ip address 131.108.1.1 255.255.0.0
!
interface serial 0
encapsulation stun
no ip address
no keepalive
stun group 1
stun route address C1 tcp 131.108.1.2
!
interface serial 1
encapsulation stun
no ip address
no keepalive
stun group 2
stun route address C1 tcp 131.108.1.2
!
```

Configuration for Router B

```
!
stun peer-name 131.108.1.2
stun protocol-group 1 sdlc
stun protocol-group 2 sdlc
!
interface ethernet 0
ip address 131.108.1.2 255.255.0.0
!
interface serial 0
encapsulation stun
no ip address
no keepalive
stun group 1
stun route address C1 tcp 131.108.1.1
!
interface serial 1
encapsulation stun
no ip address
no keepalive
stun group 2
stun route address C1 tcp 131.108.1.1
!
```

Prioritizing STUN Traffic

At times, you may wish to prioritize STUN traffic in your network over that of other protocols. The use of priority output queuing, enables this function. STUN uses a special serial line protocol called STUN for the simple serial encapsulation and TCP port 1994 for the TCP encapsulation. Therefore, to fully specify STUN traffic on priority list 4 for high priority, the following configuration is used.

Example

```
priority-list 4 stun high
priority-list 4 ip high tcp 1994
```

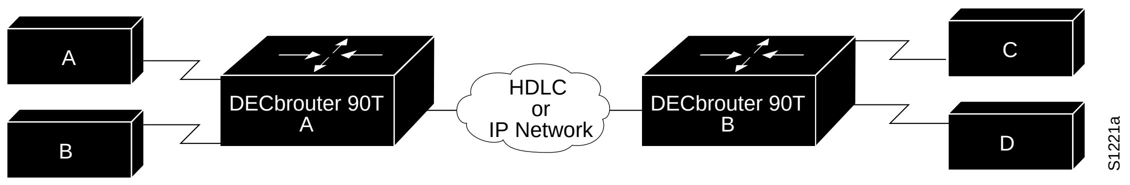
Serial Link Address Prioritization

The serial link address prioritization feature allows STUN traffic to be prioritized based on the address of a serial frame.

Figure 8–11 shows device A communicating with device C and device B communicating with device D. With the serial link address prioritization turned on, the user may choose to give A-C a higher priority over B-D across the serial tunnel.

How the serial link address prioritization is configured varies slightly depending on whether the STUN uses the HDLC encapsulation or TCP/IP encapsulation. The first step, however, is always to configure the STUN priority list.

Figure 8–11 Serial Link Address Prioritization



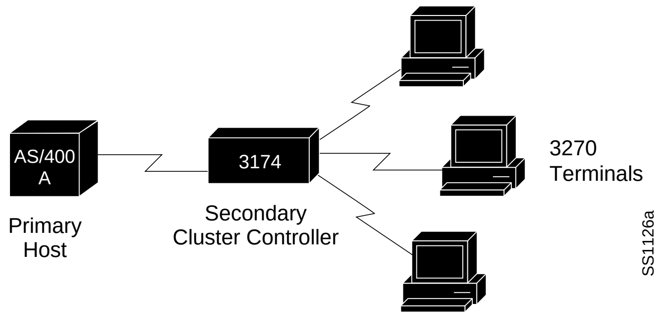
Using Proxy Polling in SDLC Environments

In normal communication between an SDLC primary node and its secondary node, the secondary node is only allowed to send data to the primary node in response to a poll from the primary node. In Figure 8–12, an AS/400 host is attached to a 3174 controller that handles transfers from several attached 3270 terminals.

Note

The proxy polling feature is provided for compatibility with prior software releases. This functions provided by proxy polling have been enhanced and superseded by the SDLC STUN with TCP/IP encapsulation and Local Acknowledgment features. Use of proxy polling is not recommended.

Figure 8–12 Typical IBM SDLC Primary and Secondary Node Configuration



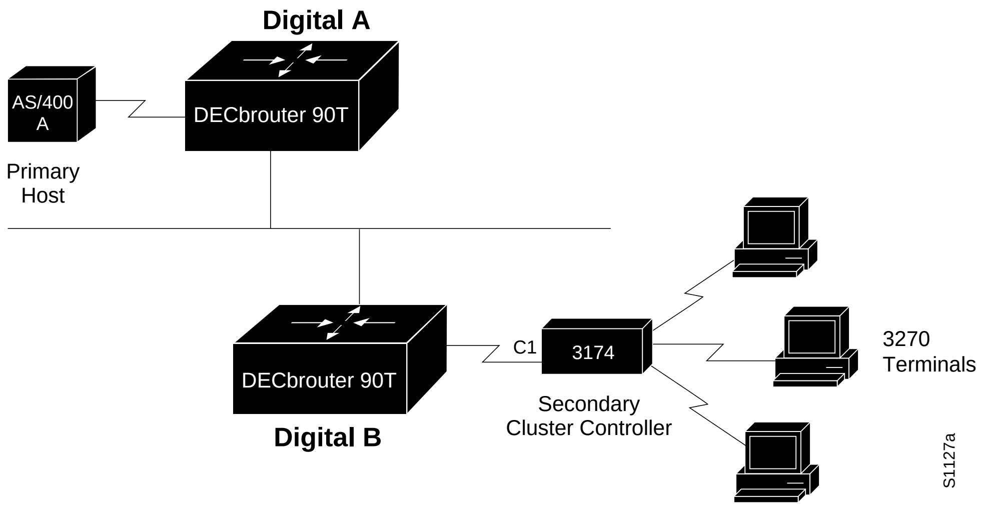
If one of the 3270-style terminals attached to the 3174 controller wants to initiate a data transfer, (usually the result of a user at the terminal pressing the Enter key), the 3174 device would not be able to immediately transfer the data back to the host AS/400 because the 3174 is the secondary node on the link. Instead, it must hold the data until a poll is received from the AS/400, which is the primary node in this example. Once the poll is received, the data can be transmitted.

The primary host ensures a reasonable response time for its secondary nodes by sending out polls at a rate that is often more than 20 times per second. With two devices sharing a single, dedicated serial line, as in the preceding example, this poses no problem, because the link would be idle without the polls.

In the example illustrated in Figure 8–13, the frequent polls and their replies would constantly travel between the two routers across the shared Ethernet.

Such constant traffic can create bottlenecks and loads that may not be appropriately handled.

Figure 8–13 IBM SDLC Primary and Secondary Nodes with Proxy Polling Feature



Configuring Proxy Polling

The proxy polling feature alleviates the load across the network by allowing the routers to act as proxies for the primary and secondary nodes, thus keeping polling traffic off of the shared links.

With proxy polling enabled, the router labeled A in Figure 8–13 would reply to the AS/400 poll requests as a proxy for the secondary node, thereby keeping the polls and requests off of the shared Ethernet. Similarly, the router labeled B would act as a proxy for the primary node and periodically send polls to the secondary 3174 device, thereby keeping its replies off of the shared cable. Only significant information is passed across the shared Ethernet.

Enabling Proxy Polling

Use these interface subcommands to enable proxy polling:

stun proxy-poll address *address* **modulus** *modulus*
{**primary** | **secondary**}
no stun proxy-poll address *address* **modulus** *modulus*
{**primary** | **secondary**}

stun proxy-poll address *address* **discovery**
no stun proxy-poll address *address* **discovery**

Enter the address of the device on which to enable proxy polling with the *address* argument.

Enter the modulus of the link as defined by the MODULUS parameter specified in the line descriptions on your SDLC host with the *modulus* argument. (This most often will be eight.)

Use the **primary** or **secondary** keyword to indicate which role the SDLC device is playing.

Use the command form with the **discovery** keyword when you do not want to specify the primary and secondary ends and the modulus used on an SDLC link, or when such connections are negotiable.

Use of the **discovery** keyword is not recommended except where you cannot avoid end hosts that negotiate the status, because proxying will be disabled on the link until session start-up, when the appropriate primary and secondary status, as well as modulus on the link, can be discovered. Until this time, all polls travel through the network.

By default, proxy polling is disabled. Once enabled, use the NO STUN PROXY-POLL command with the appropriate arguments to return to the default state.

Example

This command enables proxy polling for a secondary device at address C1 on interface serial 2 running with modulus 8:

```
!
interface serial 2
stun proxy-poll address C1 modulus 8 secondary
!
```

Configuring a Proxy Poll Interval

You can change the number of milliseconds between each sequence of proxy polls generated on the secondary side of a connection. Use the global configuration command STUN POLL-INTERVAL to do so. The command has this syntax:

stun poll-interval *milliseconds*
no stun poll-interval

The command applies to all secondary proxy sessions in the router.

Enter the number of milliseconds desired with the *milliseconds* argument. The default and minimum value that can be specified is 20, or 1/50th of a second.

The NO STUN POLL-INTERVAL command returns the default state.

Example

This example sets the poll interval to 100 milliseconds, or 10 times per second:

```
!
stun poll-interval 100
!
```

Configuring a Primary Side Pass-Through Interval

Periodically, even when proxy polling is enabled, the router on the primary side of an SDLC connection will pass through a poll from the primary SDLC device through the network to the secondary SDLC device. This action causes the secondary device's reply to also traverse the entire network. This periodic pass-through provides an insurance mechanism that makes sure the primary SDLC device maintains an accurate notion of the secondary SDLC device status.

You can change the number of seconds between each pass-through of polls between the primary and secondary SDLC devices. Use the global configuration command STUN PRIMARY-PASS-THROUGH to do so. The full command syntax follows.

stun primary-pass-through *seconds*
no stun primary-pass-through

The STUN PRIMARY-PASS-THROUGH command applies to all primary proxy sessions in the router. Use the NO STUN PRIMARY-PASS-THROUGH command to return to the default state of 20 seconds.

Example

This example sets the primary pass-through interval to 30 seconds.

```
!
stun primary-pass-through 30
!
```

Defining Your Own STUN Protocols

The STUN implementation allows you to define your own STUN protocols. This step must be done before defining the protocol group using the STUN PROTOCOL-GROUP command.

This feature allows you to transport any serial protocol that meets the following criteria across a DECbrouter 90 router internetwork.

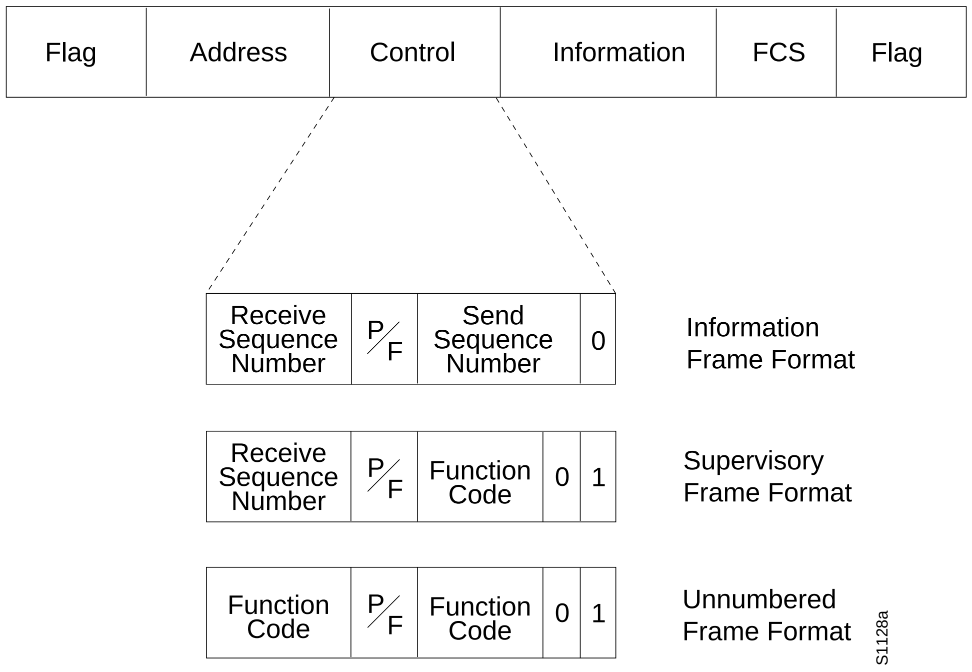
- The protocol uses full-duplex conventions (RTS/CTS always high).
- The protocol uses standard HDLC checksums and framing (beginning/end of frames, data between frames).

In addition, if you want to use anything but the predefined, basic protocol that does not allow the specification and routing by an address to achieve a virtual multidrop result, your protocol also must meet the following constraints:

- Addresses are contained in a constant location (offset) within the frame.
- Addresses are found on a byte boundary.

As an overview, SDLC frames have two formats, basic and extended. In the basic SDLC frame shown in Figure 8–14, the Address and Control fields are both 8 bits (also one byte or octet) wide. In the Extended Control format, the Control field is 16 bits wide.

Figure 8–14 SDLC Frame Format



The Address field contains the address of a sending or receiving station, depending upon the procedure. The Control field contains information that determines the type of SDLC frame being transmitted, either:

- Information frames used for data transfer
- Supervisory frames that control the flow of data
- Unnumbered frames that control the link

The Frame Check Sequence (FCS) field is always 16 bits long. A Flag is a special bit sequence that defines the beginning and end of a frame. The bytes in the SDLC frame, except for the FCS, are always transmitted low-order bit first. The FCS bits are transmitted high-order bit first.

Use the STUN SCHEMA global configuration command to specify the format, or schema, for your protocol. The command syntax follows.

stun schema *name* **offset** *constant-offset* **length** *address-length* **format** *format-keyword*
no stun schema *name*

The argument *name* is a name that defines your protocol. It can be up to 20 characters in length.

The argument *constant-offset* specifies the constant offset, in bytes, for the address to be found in the frame. The argument *address-length* specifies the length of that address in bytes. The length is limited. These limits are described in Table 8–5.

Table 8–5 Address Length Limitations

If format-keyword is:	the address-length is:
decimal	4
hexadecimal	8
octal	4

The argument *format-keyword* specifies the format to be used to specify and display addresses for routes on interfaces that use this STUN protocol. The allowable formats are listed in Table 8–6.

Table 8–6 Allowable Schema Formats

Format	Allowable Base
decimal	base 10 addresses (0-9)
hexadecimal	base 16 addresses (0-f)
octal	base 8 addresses (0-7)

The command NO STUN SCHEMA removes the schema.

Example

This command defines a format of SDLC as a new STUN protocol. (This definition of SDLC would not support the proxy polling option available with the predefined protocol definition.)

```
!
stun schema new-sdlc offset 0 length 1 format hexadecimal
!
```

Once defined, new protocols can be used in the STUN PROTOCOL-GROUP interface subcommands to tie STUN groups to the new protocols.

Monitoring STUN

This section describes the EXEC commands you use to monitor the state of the STUN.

Displaying the Current Status of STUN

Use the SHOW STUN command to examine the current status of the STUN. The command has this format:

show stun

Sample output is as follows:

```
DigitalA# show stun
This peer: 131.108.10.1
Serial0 -- 3174 Controller for test lab (group 1 [sdlc])
 state  rx_pkts  tx_pkts  drops  poll
  7[ 1] IF Serial1        open     20334    86440      5  8P
 10[ 1] TCP 131.108.8.1   open      6771     7331      0
all[ 1] TCP 131.108.8.1   open    612301  2338550   1005
```

The first entry reports that proxy polling is enabled for address 7 and that

Serial 0 is running with modulus 8 on the primary side of the link. The link has received 20,334 packets, transmitted 86,440 packets, and dropped 5 packets. See Table 23-7.

Table 8–7 STUN Status Display Field Descriptions

Field	Descriptions
This peer	Lists the peer-name or address. The interface name (as defined by the interface DESCRIPTION subcommand), its STUN group number, and the protocol associated with the group are shown on the header line.
STUN address	Address or the word "all" if the default forwarding entry is specified, followed by a repeat of the group number given for the interface.
Type of link	Description of link, either a serial interface using Serial Transport ("IF" followed by interface name), or a TCP connection to a remote router ("TCP" followed by IP address).
state	State of the link: open is the normal, working state; direct indicates a direct link to another line, as specified with the **direct** keyword on the STUN ROUTE interface subcommand.
rx_pkts	Number of received packets.
tx_pkts	Number of transmitted packets.
drops	Number of packets that for whatever reason had to be dropped.
poll	Report of the proxy poll parameters, if any. A "P" indicates a primary and an "S' indicates a secondary node. The number before the letter is the modulus of the link.

Displaying the Proxy States

Use the SHOW STUN SDLC command to examine the proxy state of various interfaces on an address-by-address basis. The command has this format:

show stun sdlc

Sample output is as follows:

```
DigitalA# show stun sdlc
Serial 1 -- 3174 controller for test lab
B3: s2 C1: p4 DE: d6
```

This example output shows that Serial 1 has three addresses for which proxy polling is enabled. These are B3, for which this end is a secondary link in state 2; C1 for which this end is a primary link and in state 4; and DE, which is in the primary/secondary/modulus in discovery state 6.

The display reports the status of the interfaces using SDLC encapsulation and tells whether proxy polling is enabled for that interface. Interfaces with proxy polling enabled are noted with the address followed by an indication of node type, "s" for secondary, "p" for primary, or "d" for discovery and the state of the node. Possible node states are defined in Table 8–8.

Table 8–8 Node States

State	Description
0	Significant data traveling between the primary and secondary node. No proxy polling occurring.
1-3	Proxy polling in process of being initiated.
4	Proxy polling is activated for the link at this time. If this in the primary node, the router responds to the primary's poll. If this is the secondary node, the router will periodically generate polls for potential response by the secondary.
5	The primary router has data from the secondary to send to the primary SDLC machine, but is waiting for a poll from that machine before transmitting the data.
6-7	This router still is trying to determine the primary/secondary character and modulus of the SDLC hosts attached to it. No proxying is done in these states.

Debugging STUN

This section describes the privileged EXEC debugging commands you use to debug operation of the STUN. Generally, you will enter these commands during troubleshooting sessions with Digital staff. For each DEBUG command, there is a corresponding UNDEBUG command to disable the reports.

debug stun-packet [*group*] [*address*]

The DEBUG STUN-PACKET command enables debugging of packets traveling through the STUN links. When enabled, it will produce numerous messages showing every packet traveling through the links. The optional argument *group* is the decimal integer assigned to a group. When the *group* parameter is given, output will be limited to only packets associated with STUN group specified. If the optional *address* argument is also specified, output will be further limited to only those packets with the STUN address specified. The *address* argument is in the format appropriate for the STUN protocol running for the group for which it is specified.

debug stun

The DEBUG STUN command enables debugging of STUN connections and status. When enabled, it will cause messages showing connection establishment and other overall status message to be displayed.

STUN Global Configuration Command Summary

Following are the global configuration commands used to configure the STUN function. These commands can appear anywhere in the configuration file.

[**no**] **locaddr-priority-list** *list address-number queue-keyword*

Establish queuing priorities based on the address of the logical unit. The argument *list* is an arbitrary integer between 1 and 10 that identifies the LU address priority list selected by the user. The argument *address-number* is the value of the LOCADDR= parameter on the LU macro, which is a one-byte address of the LU in hex. The argument *queue-keyword* is a priority queue name, one of high, medium, normal, or low.

[**no**] **priority-list** *list* **stun** *queue-keyword address group-number address-number*

Establishes queuing priorities based on the address of the serial link on a STUN connection. The argument list is an arbitrary integer between 1 and 10 that identifies the priority list selected by the user. The keyword **stun** indicates that this is a serial tunnel feature. The argument *queue-keyword* is a priority queue name, one of high, medium, normal, or low. The keyword address is used with the two arguments *group-number* and *address-number* that uniquely identify a STUN serial link. The argument *group-number* is the group number used in the STUN GROUP interface configuration subcommand. The argument *address-number* is the address of the serial link.

[**no**] **stun cos-enable**

Enables SNA class-of-service, which allows prioritization of FEP traffic.

[**no**] **stun peer-name** *ip-address*

Enables or disables STUN. The argument *ip-address* is the IP address by which this STUN peer is known to other STUN peers that are using TCP/IP encapsulation.

[**no**] **stun poll-interval** *milliseconds*

Changes the interval between each sequence of proxy polls generated on the secondary side of the connection. The argument *milliseconds* is the interval desired, in milliseconds. Default and minimum value is 20, or 1/50th of a second, which is returned by use of the **no** keyword.

[**no**] **stun primary-pass-through** *seconds*

Changes the number of seconds between each pass through of polls between the primary and secondary SDLC devices. The argument *seconds* defines the interval. The no version of the command restores the default value of 20 seconds.

[**no**] **stun protocol-group** *group-number protocol-keyword* **sdlc-tg**

Associates or removes group numbers with protocol names. The *group-number* argument can be any number you select between 1 and 255. The *protocol-keyword* argument is any predefined protocol, or protocol you define. Any STUN connections that are members of an **sdlc-tg** group must be locally acknowledged; that is, **local-ack** must be specified as a parameter on the STUN ROUTE interface command. The **no** version of the command with the appropriate group number and protocol removes an interface from the group.

[**no**] **stun schema** *name* **offset** *constant-offset* **length** *address-length* **format** *format-keyword*

Specifies or removes a format, or schema, for a user-defined protocol. The argument *name* defines the protocol. The argument *constant-offset* specifies the constant offset, in bytes, for the address to be found in the frame. The argument *address-length* specifies the length of that offset. The argument *format-keyword* specifies the format to be used to specify and display addresses for routes on interfaces that use this STUN protocol. The NO STUN SCHEMA *name* command removes the schema. The allowable formats and their maximum lengths are described in Table 8–9.

Table 8–9 Allowable Schema Formats

Formats	Base	Length
decimal	base 10 addresses (0-9)	4
hexadecimal	base 16 addresses (0-f)	8
octal	base 8 addresses (0-7)	4

STUN Interface Subcommand Summary

Following are the interface subcommands used to configure STUN. These commands follow an INTERFACE command.

encapsulation stun

Enables the STUN function. This command must be specified in order to use STUN.

[**no**] **locaddr-priority** *list*

Assigns local address priority list to the output interface. The argument *list* is the priority list number of the input interface.

[**no**] **priority-group** *list*

Assigns a priority group to the output interface. The argument *list* is the priority list number of the output interface.

[**no**] **stun group** *group-number*

Places STUN-enabled interface in a previously defined group. The argument *group-number* is user-defined and must be between 1 and 255 (decimal).

[**no**] **stun proxy-poll address** *address* **modulus** *modulus*
{**primary** | **secondary**}
[**no**] **stun proxy-poll address** *address* **discovery**

Enables or disables proxy polling.

The *address* argument is the address of the device on which to enable proxy polling. The *modulus* argument is the modulus of the link as defined by the MODULUS parameter specified in the line descriptions on the SDLC host.

The **primary** or **secondary** keyword indicate which role the SDLC device is playing.

The **discovery** keyword is used when primary and secondary ends are not specified and connections are negotiated.

Default is proxy polling disabled.

[**no**] **stun route all tcp** *ip-address*
[**no**] **stun route all interface serial** *interface-number*
[**no**] **stun route all interface serial** *interface-number* **direct**
[**no**] **stun route address** *address-number tcp ip-address* [**local-ack**][**priority**]
[**no**] **stun route address** *address-number interface* **serial** *interface-number*
[**no**] **stun route address** *address-number interface* **serial** *interface-number*
direct

Enables or disables forwarding of frames on the interface.

The **all** keyword is used when all STUN traffic received on the input interface will be propagated regardless of what address is contained in the SDLC frame.

The **tcp** keyword causes the TCP/IP encapsulation to be used to propagate frames that match the entry. Enter the address that identifies the remote STUN peer that is connected to the far SDLC link for the *ip-address* argument.

The **local-ack** keyword specifies that SDLC sessions are locally terminated. The **priority** keyword may be specified along with local-ack to enable priority queuing for the SDLC frames. The prioritization feature does add overhead, so be selective in its use.

The **interface serial** keywords cause the STUN TCP/IP encapsulation function to be used to propagate the SDLC frame. There must be an appropriately configured router on the other end of the designated serial line. Enter the serial line number connected to the router for the *interface-number* argument.

The **address** keyword specifies how an SDLC frame that contains a particular address is to be propagated. The *address-number* argument varies with the protocol type. The argument will accept any octal, decimal, or hexadecimal address in the range allowed by the protocol.

The **all** and **address** keywords are used for the same input serial interface. When this is done, the address specifications take effect first. If none of these match, the **all** keyword will be used to propagate the frame.

The **direct** keyword is used to indicate that the specified interface is also a direct STUN link, rather than a serial connection to another peer.

9 LLC2 and SDLC Link-Level Support

This chapter provides an overview of DECbrouter 90's implementation of the two IBM Data Link Level protocols:

- The LLC2 protocol used in LAN media connectivity, including Token Rings
- The SDLC serial link protocol used in WAN media IBM connectivity, including serial links

LLC2 and SDLC are never used alone, but provide a data link level support for other, higher-level router features.

The following features in software Release 9.1 use LLC2:

- Local Acknowledgment
- IBM LAN Network Manager support
- SDLLC SDLC/LLC2 Media Translation
- ISO Connection-Mode Network Services (CMNS)

The feature in software Release 9.1 that uses SDLC is SDLC/LLC2 Media Translation.

Note

SDLC, in the sense described here, is not and must not be configured for the SDLC Serial Tunnel. The section on SDLC in this chapter describes situations in which the DECbrouter 90 router acts as an SDLC *station*. With Serial Tunneling, the DECbrouter 90 router does not appear as a station, but merely *passes through* the frames between two other SDLC-speaking stations.

Information on fine-tuning and customizing these protocols for use in the router is also found in this chapter. Most often, you will not need to customize these parameters, because the default system values for the parameters described in this chapter provide good results for most purposes. You will want to refer to this chapter when performance or functionality may be suffering, and a modification of these link-level parameters offers some hope in the solution of your current network problem. Also, if your network exhibits highly unusual characteristics in its very design, such as unusually large delay parameters, you will want to consult this chapter for the effects such designs can have on the operation of SDLC or LLC2.

Overview and Comparison of SDLC and LLC2

LLC2 and SDLC are both data link level, reliable protocols. Link-level protocols operate below the network level; that is, network protocols such as ISO CMNS and SNA run on top of these layers.

They are reliable because two stations communicating with LLC2 or SDLC will do the following:

- Open and close sessions with each other before and after sending data
- Acknowledge the receipt of any frames the other station sends
- Control the flow of data between them by limiting the number of frames that can be sent by one before the other acknowledges any of them
- Recover from errors and inform the other station of this

LLC2 and SDLC protocols differ primarily in two ways: the *role* each station plays in relationship to the other, and how the stations are *addressed*.

Role of LLC2 and SDLC Stations

The following describes the different roles of LLC2 and SDLC stations:

- LLC2 stations communicate in an *asynchronous balanced mode* (ABM). LLC2-speaking stations are considered *peers* and can send data to other LLC2 stations at any time.This ability to send data at any time is called being *asynchronous*. *Balanced* mode means that each station can send to others without permission.
- SDLC stations communicate in a *normal response mode* (NRM). SDLC-speaking stations assign one station to be the *primary* station and the other to be the *secondary* station. Only the primary can start and stop sessions, and the secondary can send data to the primary only in response to being *polled* by the primary. This requirement that only one of the two stations can talk at a given time is called *normal* communications. The ability for the secondary station to talk only in response to the primary station's polls is called *response mode*.

Addressing LLC2 and SDLC Stations

The following describes how LLC2 and SDLC stations are addressed:

- All frames used in LLC2 communication have both a source and destination address. LLC2-speaking stations usually communicate over a local area network (LAN) on which many stations communicate. As such, each station needs its own address. Because the LLC2 protocol runs at the link layer, the MAC (interface) address is typically used.
- SDLC frames have only a single address. SDLC-speaking stations communicate on serial links. Typically, there is one primary and one secondary station. If there were only two stations on any serial link, then no address would be needed. Any frame received would be meant for the station receiving it. However, modem- or line-sharing devices (MSDs or LSDs) are sometimes used to allow one primary station to communicate to many secondary stations. Because there is only one primary, it needs no address. However, each secondary station needs one, because each must distinguish frames meant for it versus frames meant for the others. Therefore, all SDLC frames specify a single address: the address of the secondary station.

DECbrouter 90's Implementation of LLC2

The DECbrouter 90 router supports LLC2 connections over Ethernet. LLC2 connections are used in support of the IBM LAN Network Manager, LLC2 Local Acknowledgment, SDLLC SDLC/LLC2 Media Translation, and CMNS.

All of the LLC2 commands are optional and can be changed when you want to fine-tune LLC2 performance.

Configuring LLC2

LLC2 is configured by interface. The following shows the LLC2 interface subcommand syntax for all LLC2 commands:

llc2 *keyword value*

The following list includes some instances in which you may change LLC2 parameters:

- To control the maximum number of I-frames received before being acknowledged
- To control the maximum amount of time to allow incoming I-frames to stay unacknowledged
- To control the frequency of polls
- To control the response time to poll frames

Table 9–1 shows the LLC2 parameters that you can set. For each value, the table shows the minimum and maximum allowed values, as well as the default. Following this table are descriptions of each parameter and some typical situations when you may wish to change parameter values.

Table 9–1 LLC2 Parameters

Command	Argument	Minimum	Maximum	Default
llc2 ack-max	packet-count	1	255	3
llc2 ack-delay-time	msec	1	60000	3200
llc2 idle-time	msec	1	60000	10000
llc2 local-window	packet-count	1	127	7
llc2 n2	retry-count	1	255	8
llc2 t1-time	msec	1	60000	1000
llc2 tbusy-time	msec	1	60000	9600
llc2 trej-time	msec	1	60000	3200
llc2 tpf-time	msec	1	60000	1000
llc2 xid-neg-val-time	msec	1	60000	0
llc2 xid-retry-time	msec	1	60000	60000

Note

All timers are specified in milliseconds.

The LLC2 ACK-MAX command controls the maximum number of information frames (I-frames) the router can receive before it must send an acknowledgment of these frames to their sender.

llc2 ack-max *packet-count*

As stated in the overview, an LLC2-speaking station can send only a predetermined number of frames before it must wait for an acknowledgment from the receiver. If the receiver waits until receiving a large number of frames before acknowledging any of them, and then acknowledges them all at once, it reduces overhead on the network. For example, an acknowledgment for five frames can specify that all five have been received, as opposed to sending a separate acknowledgment for each frame. Thus, the number of acknowledgments traveling on the network is reduced. To keep network overhead low, make this parameter as large as possible. However, some LLC2-speaking stations expect this to be a low number. For example, some NetBIOS-speaking stations expect an acknowledgment to every frame. Therefore, for these stations, this number is best set to 1. Experiment with this parameter to determine the best configuration.

The **llc2 ack-delay-time** controls the maximum amount of time the router allows incoming I-frames to stay unacknowledged.

llc2 ack-delay-time *msec*

Each LLC2 station, upon receiving an information frame, starts an *alarm clock*, or timer. If this alarm clock goes off before an acknowledgment for the frame has been sent, an acknowledgment will be sent for the frame, even if the **llc2 ack-max** number of received frames has not been reached. For example, assume you have the configuration that follows:

```
interface tokenring 0
llc2 ack-max 3
llc2 ack-delay-time 800
```

At time 0, for example, two information frames are received. The ack-max amount of three has not been reached, so no acknowledgment for these frames is sent. If a third frame, which would force the router to send an acknowledgment, is not received in 800 milliseconds, an acknowledgment will be sent anyway, because the ack-delay timer alarm will have gone off. At this point, because all frames are acknowledged, the counter for the ack-max purposes will be reset to zero.

Experiment with this parameter to determine the configuration that balances undesirable acknowledgment network overhead and quick response time (by receipt of timely acknowledgments) for stations sending large amounts of information frames.

The LLC2 IDLE-TIME command controls the frequency of polls during periods of idle traffic.

llc2 idle-time *msec*

Periodically, when nothing is going on in an LLC2 session (no information frames are being transmitted), LLC2 stations are likely to send an "are you there" frame to each other, to which a similar frame is sent back letting each other know that they are, in fact, still there. This parameter controls the amount of idle time that must go by before one of these "are you there" frames are sent. These "are you there" frames are actually called *Receiver Readys*.

Set this number to a low enough value to ensure a timely discovery by the router to learn whether an LLC2 station to which it is talking has died. However, you do not want to set this too low, or you will create a network overhead with too many "are you there" frames.

The LLC2 LOCAL-WINDOW command controls the maximum number of information frames the router sends before it waits for an acknowledgment to these frames.

llc2 local-window *packet-count*

Set this number to the maximum value that can be supported by the stations to which the router communicates. Setting this value too large can cause frames to be lost, because the receiving station may not be able to receive all of them.

The LLC2 N2 command controls the amount of times the router retries various operations, including sending unacknowledged frames, or retrying remote busy station polling.

llc2 n2 *retry-count*

This parameter controls the *retry-count* limit. An LLC2 station must have some limit to the number of times it will resend a frame when the receiver of that frame has not acknowledged it for any reason. After the router is told a remote station is busy, it will poll again based on the *retry-count* until it gives up. When this retry count is exceeded, the LLC2 station terminates its session with the other station.

Very often, there is little need to change the retry count limit, and doing so is not recommended.

The LLC2 T1-TIME command controls how long the router waits for an acknowledgment to transmitted I-frames.

llc2 t1-time *msec*

When an LLC2 station sends a frame, it waits for an acknowledgment from the receiver saying that this frame has been received. Of course, the sending station cannot wait forever for a response. When the frame is sent, a timer is started. This timer is called the *T1 timer* and is controlled by this parameter. If this timer reaches its limit before the acknowledgment is received, the router tries again and resends the frame.

The amount of times the router will retry sending a frame before disconnecting the session is controlled by the **llc n2** parameter.

Note

The **llc2 t1-time** parameter is extremely important, and is probably the one you will most want to change. Ensure that this parameter can account for the round-trip time between the router and its LLC2-speaking stations under heavy network loading conditions. Avoid causing unnecessary retransmissions to be sent at any cost; they will only add to the network load.

The LLC2 TBUSY-TIME command controls the amount of time the router allows the other LLC2 station to stay in a busy state before this router attempts to poll the remote station again, hoping for a clear state.

llc2 tbusy-time *msec*

LLC2 stations have the ability to tell each other that they are temporarily busy, so the other station should not attempt to send any new information frames. The frames sent to indicate this are called *Receiver Not Ready (RNR) frames*.

Usually, there is little reason to change this parameter. You would only do so to increase the value if you discover that you have LLC2-speaking stations that have unusually long busy periods before they clear their busy state indicating that they are ready to receive more I-frames. Therefore, you would not want to time these stations out.

The amount of times the router will retry polling the remote station hoping for a clear state before disconnecting the session is controlled by the **llc n2** parameter.

The LLC2 TREJ-TIME command controls the amount of time the router waits for a resend of a rejected frame before resending the reject (REJ) command to the remote.

llc2 trej-time *msec*

When an LLC2 station sends an information frame, a sequence number (called the N(S) value or *Number of Sent frame*) is included in the frame. The LLC2 station that receives these frames will expect to receive them in order. If it does not, it can reject a frame and indicate which frame it is expecting to receive instead.

Of course, the LLC2 station that sends the REJ does not want to wait forever to receive the correct frame. Therefore, upon sending a reject, it starts a *reject timer*. If the frames are not received before this timer expires, the reject is resent.

The amount of times the router retries sending the REJ frame without getting the rejected I-frame sent to it before disconnecting the session is controlled by the **llc n2** parameter.

The LLC2 TPF-TIME command controls the amount of time the router waits for a *final* response to a *poll* frame that it sent before the router resends the original *poll* frame.

llc2 tpf-time *msec*

LLC2 was designed from earlier serial link protocols, including SDLC. Therefore, when sending a command that must receive a response, or completing the transmission of data, a *poll* bit is sent in the frame. This is the receiving station's clue that the sender is expecting some response from it, be it an acknowledgment of information frames, or an acknowledgment of more administrative tasks, such as the starting and stopping of the session. Once a sender gives out the *poll* bit, it cannot send any other frame with the *poll* bit set until the receiver replies to that *poll* frame with a frame containing a *final* bit set. Obviously, if the receiver is poorly designed or buggy, it may never give that *final* bit back to the sender. Therefore, the sender could be stuck waiting for a reply that will never come. To avoid this problem, when a poll-bit-set frame is sent, a *transmit-poll-frame (TPF)* timer is started. If this timer expires, the router assumes that it can send another frame with a *poll* bit.

Usually, you will not want to change this value. If you do, it should be somewhat larger than the **t1-time**, because frames with *poll* bits can often take some time longer for a station to process once received. Increase this timer if you find that you have such slow LLC2 stations.

The **llc n2** parameter controls the number of times the router retries sending the *poll* frame to the remote station hoping for a response before disconnecting the session.

The LLC2 XID-NEG-VAL-TIME command controls the frequency of XID transmissions by the router.

llc2 xid-neg-val-time *msec*

LLC2-speaking stations can communicate *exchange of identification (XID)* frames to each other. These frames identify the stations at a higher level than the MAC address and also can contain information about the configuration of the station. These frames are typically sent only during setup and configuration periods when it is deemed that sending them is useful. The greatest frequency at which this information is transferred is controlled by this timer.

Note

Do not change the **llc2 xid-neg-val-time** parameter unless requested by Digital.

The LLC2 XID-RETRY-TIME command controls how long the router waits for a reply to the XID frames it sends to remote stations.

llc2 xid-retry-time *msec*

Along with the periodic normal retransmittal of XID frames controlled by the **xid-neg-val-time**, the amount of time that the router will wait for an XID from the remote LLC2 station in response to its XID must be controlled. This xid-retry timer controls this value. It should be set to a somewhat larger value than the T1 time, and should be set in situations where the roundtrip delay through the network is unexpectedly large.

Monitoring LLC2

The following EXEC command, with example output, gives the state of LLC2 connections within a DECbrouter 90:

show llc2

The SHOW LLC2 command shows the LLC2 connections active in the router. The following is an example of this output:

```
Ethernet0 DTE=1000.5A59.04F9,400022224444 SAP=04/04, State=NORMAL
V(S)=5, V(R)=5, Last N(R)=5, Local window=7, Remote Window=127
ack-max=3, n2=8, Next timer in 7768
xid-retry timer 0/60000 ack timer 0/1000
p timer 0/1000 idle timer 7768/10000
rej timer 0/3200 busy timer 0/9600
ack-delay timer 0/3200
CMNS Connections to:
 Address 1000.5A59.04F9 via Serial1
 Protocol is up
 Interface type X25-DCE RESTARTS 0/1
 Timers: T10 1 T11 1 T12 1 T13 1
```

Table 9–2 describes the SHOW LLC2 command fields.

Table 9–2 Show LLC2 Field Descriptions

Field	Description
Interface	Interface on which the session is established.
DTE	Address of the station to which the router is talking on this session. (The router's address is the MAC address of the interface on which the connection is established, except when Local Acknowledgment or SDLLC is used. In those cases, the address used by the router is shown as in this example, following the DTE address and separated by a comma.)
SAP	The other station's and the router's (remote/local) Service Access Point for this connection. The SAP is analogous to a "port number" on the router and allows for multiple sessions between the same two stations.
State	The current state of the LLC2 session; can be any of:
ADM	Asynchronous Disconnect Mode—A connection is not established, and either end can begin one.
SETUP	Request to begin a connection has been sent to the remote station, and this station is waiting for a response to that request.
RESET	A previously open connection has been reset because of some error by this station, and this station is waiting for a response to that reset command.
D_CONN	This station has requested a normal, expected, end of communications with the remote, and is waiting for a response to that disconnect request.
ERROR	This station has detected an error in communications and has told the other station of this. This station is waiting for a reply to its posting of this error.

(continued on next page)

Table 9–2 (Cont.) Show LLC2 Field Descriptions

Field	Description
NORMAL	Connection between the two sides is fully established, and normal communication is occurring.
BUSY	Normal communication state exists, except busy conditions on this station make it such that this station cannot receive information frames from the other station at this time.
REJECT	Out-of-sequence frame has been detected on this station, and this station has requested that the other resend this information.
AWAIT	Normal communication exists, but this station has had a timer expire and is trying to recover from it (usually by resending the frame that started the timer).
AWAIT_BUSY	A combination of the AWAIT and BUSY states.
AWAIT_REJ	A combination of the AWAIT and REJECT states.
V(S)	Sequence number of the next information frame this station will send.
V(R)	The sequence number of the next information frame this station expects to receive from the other station.
Last N (R)	Last sequence number of this station's transmitted frames acknowledged by the remote station.
Local Window	Number of frames this station can send before requiring an acknowledgment from the remote station.
Remote Window	Number of frames this station can accept from the remote.
ack-max, n2	Value of these parameters, as given in the previous configuration section.
Next timer in <nnn>	Number of milliseconds before the next timer goes off for any reason.
Timer Values	A series of timer values in the form of next-time/time-between, where "next-time" is the next time, in milliseconds, that the timer will wake, and "time-between" is the time, in milliseconds, between each timer wakeup. A "next-time" of zero indicates that the given timer is not enabled and will never wake.
CMNS parameters:	
Address	MAC address of remote station.
Protocol State	Up indicates the LLC2 and X.25 protocols are in a state where incoming and outgoing Call Requests can be made on this LLC2 connection.
Interface type	One of: X25-DCE, X25-DTE, or X25-DXE (both DTE and DCE).
RESTARTS	Restarts sent/received on this LLC2 connection.
Timers	T10, T11, T12, T13 (or T20, T21, T22, T23 for DTE); these are Request packet timers. These are similar in function to X.25 parameters of the same name.

Debugging LLC2

The following commands describe how to retrieve debugging information for LLC2. The output of these commands is detailed and contains internal program information. It can only be adequately understood when used in conjunction with Digital staff. All commands are disabled with the corresponding UNDEBUG command.

debug llc2-events

The DEBUG LLC2-EVENTS command turns on LLC2 event debugging. Actions that cause the current LLC2 state to change are displayed.

debug llc2-packets

The DEBUG LLC2-PACKETS command turns on LLC2 packets debugging. All LLC2 frames input or output to and from the router are displayed.

debug llc2-test

The DEBUG LLC2-TEST command turns on internal LLC2 code debugging.

Note

LLC2 debugging can generate large amounts of console output that can adversely affect the operation of LLC2 on the router, because packets can be delayed long enough to cause retransmissions. Only use LLC2 debugging when suggested by Digital Systems staff.

DECbrouter 90's Implementation of SDLC

This section describes the commands that apply to the SDLC implementation. They are the counterpoint to the LLC2 commands specified in the LLC2 section of this chapter.

SDLC is used in the DECbrouter 90's implementation of SDLLC, the media translator between LLC2 and SDLC and SDLC local acknowledgement.

The DECbrouter 90's implementation of SDLC supports *multipoint*, using a modem- or line-sharing device (MSD or LSD), in those configurations where the device speaks a full-duplex protocol to the router. The router can act as the primary end of the SDLC session or as a secondary station.

Configuring SDLC

The commands in this section can be used for the serial interface that supports SDLLC traffic.

Your router can act as either a primary or a secondary SDLC station for a serial line. An ENCAPSULATION SDLC-PRIMARY or ENCAPSULATION SDLC-SECONDARY command must be used to identify any interface to be configured for SDLC.

The specification of sdlc-primary emphasizes that the router is the primary station when behaving as an SDLC station.

encapsulation sdlc-primary

For example, if you are running SDLLC on interface serial 0, begin the configuration with:

```
   interface serial 0
   encapsulation sdlc-primary
```

The ENCAPSULATION SDLC-SECONDARY command is used for SDLLC configurations in which the router has established an SDLC session with an IBM machine such as a Front End Processor (FEP). The ENCAPSULATION SDLC-SECONDARY command indicates that the router is a secondary SDLC station for a serial line.

encapsulation sdlc-secondary

For example, if an FEP with a serial connection needs to communicate with a Token Ring attached to an IBM 3174 controller, the router attached to the serial FEP should be configured as a secondary SDLC station, by configuring the serial line connected to the FEP for encapsulation sdlc-secondary. The FEP is the primary station.

After the ENCAPSULATION SDLC-PRIMARY command, assign the set of secondary stations attached to the serial link by using a series of SDLC ADDRESS commands:

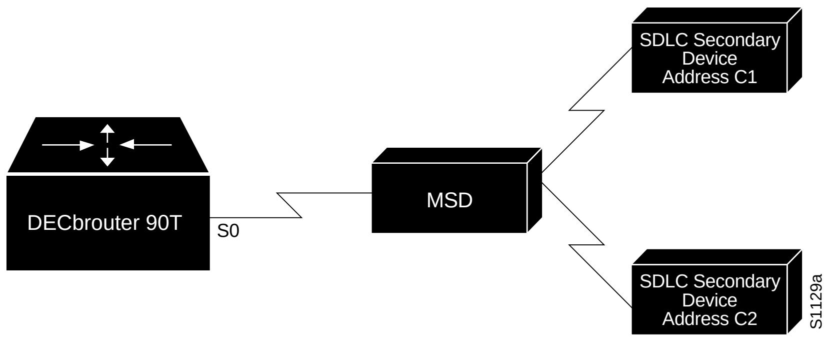
sdlc address *hexbyte*
no sdlc address

The addresses are given in hexadecimal (base 16) and are given one per line. For example, to configure interface serial 0 to have two SDLC secondary stations attached to it through a modem-sharing device (MSD) with addresses C1 and C2, begin your SDLC configuration of this interface with the following:

```
   interface serial 0
   encapsulation sdlc-primary
   sdlc address c1
   sdlc address c2
```

The network for this configuration is shown in Figure 9–1.

Figure 9–1 Two SDLC Secondary Stations Attached to a Single Serial Interface Through an MSD


SDLC Secondary Device Address C1
DECbrouter 90T
S0
MSD
SDLC Secondary Device Address C2
S1129a

SDLC is configured per interface. The following shows the SDLC command syntax for all SDLC interface subcommands:

sdlc *keyword value*

Note

All of these commands are optional and modify the behavior of SDLC on the link.

Table 9–3 shows the SDLC parameters. The command descriptions follow the table.

Table 9–3 SDLC Parameters

Command	Argument	Minimum	Maximum	Default
sdlc frmr-disable		enable/disable		disabled
sdlc t1	timeout	1	64000	3000
sdlc n1	bit-count	1	12000	12000
sdlc n2	retry-count	1	255	20
sdlc k	windowsize	1	7	7
sdlc holdq	address queue-size	1	none	12
sdlc poll-pause-timer	msec	100	10000	100
sdlc fair-poll-timer	msec	100	10000	500
sdlc poll-limit-value	count	1	10	1

The [NO] SDLC FRMR-DISABLE command is the only subcommand that has no value assigned to it.

Note

Do not change the default values for the SDLC commands unless situations warrant their change or unless Digital Systems staff suggest a change. All timers are given in milliseconds.

Controlling the SDLC Protocol Behavior

Specify the FRMR-DISABLE command to indicate that secondary stations off this link do not support *Frame Rejects* (FRMRs). The command syntax is as follows:

sdlc frmr-disable
no sdlc frmr-disable

FRMRs are *error indications* that can be sent to an SDLC station indicating that a protocol error has occurred. Not all SDLC stations support FRMRs. If this command is enabled, thereby disabling FRMRs, the router simply drops the line when it receives an error by sending a disconnect request to the remote station.

The default value for this command is to send FRMRs. This default behavior can be retrieved by specifying the **no** version of this command.

Controlling the SDLC Timers and Retry Counts

The SDLC T1 *timeout* command controls the amount of time the router waits for a reply to a frame or sequence of frames.

sdlc t1 *timeout*

When an SDLC station sends a frame, it waits for an acknowledgment from the receiver saying that this frame has been received. Of course, the sending station cannot wait forever for a response. When the frame is sent, a timer is started. For reasons of being consistent with the original specification of SDLC, this timer is called the *T1 timer* and is controlled by this parameter. If this timer reaches its limit before the acknowledgment is received, the router will try again and resend the frame.

The number of times that a retry of a frame is tried is controlled with the SDLC N2 command.

This parameter is extremely important, and is probably the one you most want to change. Ensure that this parameter can account for the round-trip time between the DECbrouter 90 and its SDLC-speaking stations under heavy network loading conditions. Avoid causing unnecessary retransmissions to be sent at any cost, because they will only add to the network load.

The SDLC N2 *retry-count* command controls the number of times the router attempts to retry an operation that has timed out.

sdlc n2 *retry-count*

The **sdlc n2** parameter controls the *retry-count* limit. When the router as an SDLC station has to resend a frame because the receiver of that frame has not acknowledged it for any reason, the SDLC station must have some limit to the number of times it will try again. When this retry count is exceeded, the SDLC station will terminate its session with the other station.

Very often, there is little need to change this parameter, and its change is not recommended.

Controlling the SDLC Frame and Window Sizes

The SDLC N1 *bit-count* command controls the maximum size of an incoming frame, in bits, on this link.

sdlc n1 *bit-count*

Frames received that exceed the specified size, in bits, are rejected, and an error condition will ensue.

The SDLC K *windowsize* command controls the router’s local send window size.

sdlc k *windowsize*

When the router is communicating with SDLC, it must have a parameter that controls the maximum number of information frames it will receive before the other end must stop sending and wait for an acknowledgment. This allows for the better creation of buffer pools and other memory objects. The **k** parameter controls this *window* of acceptable frames.

Affecting the Output Buffering Ability

The following command controls the output buffering ability.

The SDLC HOLDQ *address queue-size* command controls the maximum number of outstanding packets that can be held for pending transmission to a remote SDLC station. This is specified on a per-address basis.

sdlc holdq *address queue-size*

Use this command to increase the hold queue size from the default maximum when you find that temporary loading conditions exist that exceed the default hold queue length of the router.

Affecting Polling of the Secondary Stations

The following three parameters affect the behavior of polls in the system:

[**no**] **sdlc poll-pause-timer** *msec*
[**no**] **sdlc fair-poll-timer** *msec*
[**no**] **sdlc poll-limit-value** *count*

It is assumed that you are familiar with how polls operate on SDLC links before you attempt to change these values. A brief summary follows.

SDLC secondary stations, particularly when an MSD/LSD is used, are typically attached to the router out each of its SDLC-speaking serial stations. These stations are true SDLC secondaries, and expect the router to perform its duty as an SDLC primary station, polling them fairly and in a timely manner. As secondary stations, these stations cannot send data to the primary (the router) until they are polled for that data. Because they are sharing a single serial link, only one station can be the actively polled station at any one time.

To meet the *timely* requirement, the router must show intelligence in accepting streams of input, or in giving streams of output, to a single station, even if these streams require that more than one packet or window size of packets must be transferred. The goal of this intelligence is to keep on giving output to, or polling for, input from the same station until a full transaction of data has completely transferred in one direction.

However, meeting this requirement impacts the *fairness* to the other stations. The router, as the primary, cannot show undue favoritism to a suddenly noisy secondary and avoid its responsibility of polling the other stations in their turn. Also, the router does not want to flood the serial link with polls for otherwise idle stations. The following three parameters help control this behavior. For all three parameters, retain the defaults unless a problem is evidenced, and adjust them as little as possible until the desired behavior is obtained. Fairness is also obtained by polling stations in a round-robin manner when in an idle state.

The SDLC POLL-PAUSE-TIMER *msec* command controls how long the router pauses between sending each series of poll frames.

sdlc poll-pause-timer *msec*
no sdlc poll-pause-timer

As is typical for the primary station of an SDLC connection, the router generates polls periodically to each of the secondary stations to solicit their input. After a full cycle of polls to all stations off a single serial interface are sent and replied to, and there is no output to give to any secondary station from the primary station, the router will pause and let the line remain idle for the amount of this timer before beginning to poll the stations again.

Because the secondaries cannot transmit data (such as a user pressing an Enter key) until they are polled, increasing this timer can increase response time to the users. However, making this parameter too small can flood the serial link with unneeded polls and require the secondary stations to spend wasted CPU time processing them.

The default value is 100 msec. This default value can be retrieved by specifying the no version of this command.

The SDLC FAIR-POLL-TIMER *msec* command controls the maximum time that output can be sent to any secondary before polling for input must begin.

sdlc fair-poll-timer *msec*
no sdlc fair-poll-timer

As is typical for the primary station of an SDLC connection, output from the primary to any of the secondary stations is given a higher priority than receiving input from any of them. The **fair-poll-timer** indicates the maximum time that can elapse before output from the primary is no longer given that priority and the next poll of the next secondary station is attempted.

Increasing this parameter allows screen updates to appear smoother, but has the side effect of possibly increasing, for example, the time it takes for users pressing an Enter key to have their data sent to their station.

The default value is 500 msec. This default value can be retrieved by specifying the no version of this command.

The SDLC POLL-LIMIT-VALUE *count* command controls how many times a single secondary station can be polled for input before the next station in the poll list must be polled.

sdlc poll-limit-value *count*
no sdlc poll-limit-value

As is typical for the primary station of an SDLC connection, if a secondary station sends its full possible window of input to the primary router, the router immediately will repoll this same secondary for more data in an attempt to capture the complete transaction at one time. The **poll-limit-value** indicates how many times this can happen before the next station in the poll loop must be polled.

Increasing the value allows for smoother transaction processing but can delay polling of other stations or giving output to other stations.

The default value is 1. This default value can be retrieved by specifying the **no** version of this command.

Monitoring SDLC

The following command displays the SDLC information for a given interface:

show interfaces *type unit*

For example, assume that interface Serial 0 is supporting the SDLLC function, and as such is configured as an SDLC primary interface. The following output would result from the SHOW INTERFACES command:

```
Serial 0 is up, line protocol is up
  Hardware is MCI Serial
  MTU 1500 bytes, BW 1544 Kbit, DLY 20000 usec, rely 255/255, load 1/255
  Encapsulation SDLC-PRIMARY, loopback not set
      Timers (msec): poll pause 100 fair poll 500. Poll limit 1
      [T1 3000, N1 12016, N2 20, K 7] timer: 56608 Last polled device: none
      SDLLC [ma: 0000.0C01.14--, ring: 7 bridge: 1, target ring: 10
             largest token ring frame 2052]
  SDLC addr C1 state is CONNECT
      VS 6, VR 3, RCNT 0, Remote VR 6, Current retransmit count 0
      Hold queue: 0/12 IFRAMEs 77/22 RNRs 0/0 SNRMs 1/0 DISCs 0/0
      Poll: clear, Poll count: 0, chain: p: C1 n: C1
      SDLLC [largest SDLC frame: 265, XID: disabled]
  Last input 00:00:02, output 00:00:01, output hang never
  Output queue 0/40, 0 drops; input queue 0/75, 0 drops
  Five minute input rate 517 bits/sec, 30 packets/sec
  Five minute output rate 672 bits/sec, 20 packets/sec
     357 packets input, 28382 bytes, 0 no buffer
     Received 0 broadcasts, 0 runts, 0 giants
     0 input errors, 0 CRC, 0 frame, 0 overrun, 0 ignored, 0 abort
     926 packets output, 77274 bytes, 0 underruns
     0 output errors, 0 collisions, 0 interface resets, 0 restarts
     2 carrier transitions
```

Table 9–4 shows the fields relevant to all SDLC connections on this interface.

Table 9–4 Show Interfaces Fields and Descriptions

Parameter	Description
Timers (msec)	
poll pause, fair poll, poll limit	Current values of these timers as described in the configuration section for this interface.
T1, N1, N2, K	Values for these parameters as described in the configuration section for this interface.

Table 24-5 shows other data given for each SDLC secondary configured to be attached to this interface.

Table 9–5 SDLC Secondary Descriptions

SDLC Secondary	Description
addr	Address of this secondary.
State	Current state of this connection which are any of:
DISCONNECT	No communication is being attempted to this secondary.
CONNECT	A normal connect state exists between this router and this secondary.
DISSCENT	This router has sent a disconnect request to this secondary and is awaiting its response.
SNRMSENT	This router has sent a connect request (SNRM) to this secondary and is awaiting its response.
THEMBUSY	This secondary has told this router that it is temporarily unable to receive any more information frames.
USBUSY	This router has told this secondary that it is temporarily unable to receive any more information frames.
BOTHBUSY	Both sides have told each other that they are temporarily unable to receive any more information frames.
ERROR	This router has detected an error and is waiting for a response from the secondary acknowledging it.
VS	Sequence number of the next information frame this station sends.
VR	Sequence number of the next information frame from this secondary that this station expects to receive.
Remote VR	Last frame transmitted by this station that has been acknowledged by the other station.
Current retransmit count:	How many times the current I-frame or sequence of I-frames has been retransmitted.
Hold Queue	Number of frames in hold queue/maximum size of hold queue.
IFRAMEs, RNRs, SNRMs, DISCs	Sent/received count for these frames.
Poll	"Set" if this router has a poll outstanding to the secondary; "clear" if it does not.
Poll Count	Number of polls in a row that have been given to this secondary at this time.
Chain	Shows the previous (p) and next (n) secondary address on this interface in the *round robin loop* of polled devices.

Debugging SDLC

debug sdlc

This command displays SDLC frames received and sent by any DECbrouter 90 serial interface involved in supporting SDLC end station functions. The debugging output can be disabled with the UNDEBUG command.

LLC2 Interface Subcommand Summary

The following is an alphabetical list of the LLC2 interface subcommands.

llc2 ack-delay-time *msec*

Controls the maximum amount of time the router allows incoming I-frames to stay unacknowledged. Each LLC2 station, upon receiving an information frame, starts an *alarm clock*, or timer. If this alarm clock goes off before an acknowledgment for the frame has been sent, an acknowledgment will be sent for the frame, even if the **llc2 ack-max** number of received frames has not been reached.

llc2 ack-max *packet-count*

Controls the maximum number of information frames (I-frames) received by the router before it must send an acknowledgment to these frames to their sender. Experiment with this parameter to determine the best configuration.

llc2 idle-time *msec*

Controls the frequency of polls during periods of idle traffic.

llc2 local-window *packet-count*

Controls the maximum number of information frames sent by the router before it waits for an acknowledgment to these frames.

llc2 n2 *retry-count*

Controls the amount of times the router retries various operations, including sending unacknowledged frames, or retrying the polling of a remote busy station.

llc2 t1-time *msec*

Controls how long the router waits for an acknowledgment to transmitted I-frames. The **llc2 t1-time** parameter is extremely important, and is probably the one you most want to change.

llc2 tbusy-time *msec*

Controls the amount of time the router allows the other LLC2 station to stay in a busy state before this router attempts to poll the remote station again hoping for a clear state.

llc2 tpf-time *msec*

Controls the amount of time the router waits for a *final* response to a *poll* frame that it sent before the router resends the original *poll* frame.

llc2 trej-time *msec*

Controls the amount of time the router waits for a resend of a rejected frame before resending the reject (REJ) command to the remote.

llc2 xid-neg-val-time *msec*

Controls the frequency of XID transmissions by the router. Do not change the **llc2 xid-neg-val-time** parameter unless requested by Digital.

llc2 xid-retry-time *msec*

Controls how long the router waits for a reply to the XID frames it sends to remote stations.

SDLC Interface Subcommand Summary

The following is an alphabetical list of the SDLC interface subcommands.

encapsulation sdlc-primary

Identifies any interface that requires an SDLC configuration. This command always must be given, no matter what functionality is using the SDLC support. The specification of **sdlc-primary** emphasizes that the router is always the primary station when behaving as an SDLC station.

encapsulation sdlc-secondary

Indicates that the router is a secondary SDLC station for a serial line. The encapsulation SDLC-SECONDARY command is used for SDLLC configurations in which the router has an SDLC session with an IBM machine such as a Front End Processor (FEP).

[**no**] **sdlc address** *hexbyte*

Assigns the set of secondary stations attached to the serial link using a sequence of SDLC ADDRESS commands.

[**no**] **sdlc fair-poll-timer** *msec*

Controls the maximum time that output can be sent to any secondary before polling for input must begin. The **fair-poll-timer** indicates the maximum time that can elapse before output from the primary is no longer given that priority and the next poll of the next secondary station is attempted. The default value is 500 msec. This default value can be retrieved by specifying the **no** version of this command.

[**no**] **sdlc frmr-disable**

Indicates that secondary stations off this link do not support *Frame Rejects* (FRMRs). If this command is enabled, thereby disabling FRMRs, the router simply drops the line when it receives an error by sending a disconnect request to the remote station. The default value for this command is to send FRMRs. This default behavior can be retrieved by specifying the **no** version of this command.

sdlc holdq *address queue-size*

Controls the maximum number of outstanding packets that can be held for pending transmission to a remote SDLC station. This is specified on a per-address basis.

This command is specified for each SDLC address, and thereby, each SDLC secondary station on the serial link. Use this command to increase the hold queue size from the default maximum when you find that there are temporary loading conditions that exceed the default hold queue length of the router.

sdlc k *windowsize*

Controls the router's local send window size. The **k** parameter controls this *window* of acceptable frames.

sdlc n1 *bit-count*

Controls the maximum size of an incoming frame, in bits, on this link. Frames received that exceed the specified size, in bits, are rejected, and an error condition will ensue.

sdlc n2 *retry-count*

Controls the number of times the router attempts to retry an operation that has timed out. When this retry count is exceeded, the SDLC station will terminate its session with the other station.

[**no**] **sdlc poll-limit-value** *count*

Controls how many times a single secondary station can be polled for input before the next station in the poll list must be polled. The **poll-limit-value** indicates how many times this can happen before the next station in the poll loop must be polled. The default value is 1. This default value can be retrieved by specifying the **no** version of this command.

[**no**] **sdlc poll-pause-timer** *msec*

Controls how long the router pauses between sending each series of poll frames.The default value is 100 msec. This default value can be retrieved by specifying the **no** version of this command.

sdlc t1 *timeout*

Controls the amount of time the router waits for a reply to a frame or sequence of frames.

10 Fast Switching

There are a variety of Release 9.14 protocols and interfaces that support fast switching. Use fast switching to obtain faster packet transfer on all connections except slow serial links (56k and below).

For information about configuring fast switching of protocols, refer to the appropriate routing protocol chapter in the *DECbrouter 90 Products Configuration and Reference* publication.

Table 10–1 is a matrix of the Release 9.14 protocols and interfaces that support fast switching.

Table 10–1 Release 9.14 Fast Switching

Protocol/Interface	Ethernet	Serial
IP	yes	yes
IPX	yes	yes
XNS	yes	yes
DECnet	yes	yes
CLNS	yes	yes
Appletalk	no	no
Transparent Bridging	yes	yes
Translational bridging	no	no
Encapsulated bridging	n/a	n/a
Source-route bridging	n/a	n/a
Remote source-route bridging	yes	yes
FST	yes	yes

11
Point-to-Point Protocol (PPP) for DECnet IV and OSI

PPP supports DECnet IV, OSI, and IP on the DECbrouter 90T running Software Release 9.14. If an interface is configured for PPP encapsulation and DECnet or OSI routing, PPP will attempt to negotiate the network control protocol for DECnet or OSI.

This implementation of PPP is in compliance with RFC 1331. This implementation of DECnet conforms to RFC 1376 entitled "The PPP DECnet Phase IV Control Protocol" by Steven Senum. This implementation of OSI conforms to RFC 1377, "The PPP OSI Network Layer Control Protocol" by David Katz.

The following topics are covered in this chapter:

- Configuring the Point-to-Point Protocol (PPP)
- Enabling DECnet routing
- Enabling OSI routing

There is a command summary at the end of the chapter.

Configuring the Point-to-Point Protocol (PPP)

Point-to-Point Protocol (PPP), described in RFC 1331, is designed to encapsulate Internet Protocol (IP), DECnet IV, and OSI datagrams and other network layer protocol information over point-to-point links. RFC 1331, "The Point-to-Point Protocol (PPP) for the Transmission of Multiprotocol Datagrams over Point-to-Point Links" by William Simpson, defines the set of options that are negotiated during startup.

IP, DECnet IV, and OSI are the supported upper-level protocols. These upper-level protocols can be sent or received over a point-to-point link using PPP if the upper-level protocol is enabled on an interface. Refer to RFC 1331 for definitions of the codes and protocol states.

Use the following interface subcommand to enable PPP on an interface:

```
encapsulation ppp
```

Example

The following example enables PPP encapsulation on interface serial 0.

```
interface serial 0
encapsulation ppp
```

Implementing Protocols on Systems Running PPP

Protocols and Network Control Programs (NCPs) configured after PPP is up and running will be implemented after the interface is disabled by the SHUTDOWN command and then brought up again by the NO SHUTDOWN command. See the *DECbrouter 90 Products Configuration and Reference, Volume 1* publication for more information about the SHUTDOWN command.

Monitoring Line Status

Use the KEEPALIVE command in the *DECbrouter 90 Products Configuration and Reference, Volume 2* to monitor line status.

Enabling DECnet Routing

To enable or disable DECnet routing, use the following global configuration command:

decnet routing *decnet-address*
NO decnet routing

The argument *decnet-address* is an address in DECnet format X.Y, where X is the area number and Y is the node number. There is no default router address; you must specify this parameter for DECnet operation.

Example

In this example, DECnet routing is enabled for the router in area 21 with node number 456:

```
decnet routing 21.456
```

Note

Enabling DECnet changes the MAC addresses of the router's interfaces. This is not a problem on routers equipped with nonvolatile memory. On systems that attempt to get their IP network addresses from network servers rather than from nonvolatile memory, changes in the hardware address can confuse other IP-speaking hosts. If you are attempting to use DECnet on such a configuration, be sure to set all global DECnet parameters before enabling DECnet routing on the interfaces.

Assigning the Cost

After DECnet routing has been enabled, you must assign a cost to each interface over which you want DECnet to run. Assigning a cost enables DECnet routing for an interface. Most DECnet installations have an individualized routing strategy for using costs. Therefore, check the routing strategy used at your installation to ensure that the costs you specify are consistent with those set for other hosts on the network.

Use the DECNET COST interface subcommand to set a cost value for an interface and enable DECnet routing:

> **decnet cost** *cost*
> **no decnet cost**

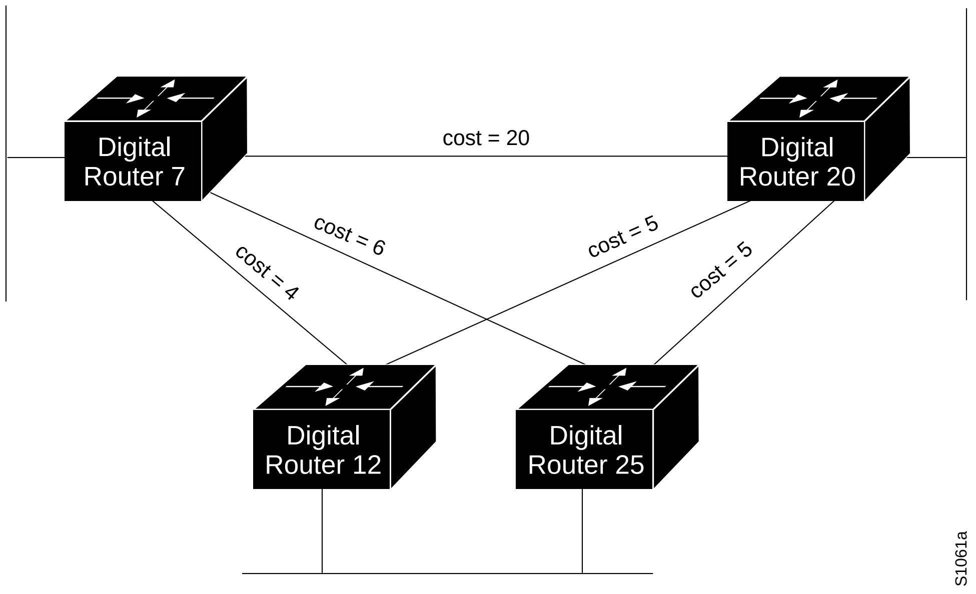
The argument cost is an integer from 1 to 63. There is no default cost for an interface, although suggested costs are 4 for Ethernet, and greater than 10 for serial links. Use the NO DECNET COST subcommand to disable DECnet routing for an interface.

Example

The following example establishes a DECnet routing process for the router at 21.456, then sets a cost of four for the serial 0 interface:

```
decnet routing 21.456
interface serial 0
encapsulation ppp
decnet cost 4
```

Figure 11–1 DECnet Cost Values/FIG_V3_DEnt



Enabling OSI Routing

Connectionless Network Protocol/Connectionless Network Service (CLNS) is an OSI network-layer protocol. To implement OSI routing, configure the system and interface for CLNS routing.

Use the following command to enable CLNS routing:

```
clns routing
no clns routing
```

Use the NO CLNS ROUTING command to disable CLNS routing.

You need to decide now whether you want to use static or dynamic routing. Static routing is required if you must interoperate with another vendor's equipment that does not support IS-IS. In addition, static routing is generally preferable for operation over X.25, Frame Relay, or SMDS networks.

ISO CLNS is enabled automatically when you use the following commands to configure IS-IS or ISO-IGRP routing on an interface: CLNS ROUTER ISIS, IP ROUTER ISIS, and CLNS ROUTER ISO-IGRP. However, if you do not intend to perform any dynamic routing on an interface, but want your router to pass ISO CLNS packet traffic to end systems, enable ISO CLNS for an interface by using the following interface subcommand:

```
clns enable
no clns enable
```

Use the NO CLNS ENABLE command to disable ISO CLNS on a particular interface.

For information about configuring OSI routing protocols, see the *DECbrouter 90 Products Configuration and Reference, Volume 3* publication.

Example

In the following example, OSI routing is globally configured, and PPP encapsulation and OSI routing are enabled on interface serial 0:

```
! Globally configure OSI routing
clns routing
interface serial 0
! Implement PPP encapsulation on interface serial 0
encapsulation ppp
! Enable OSI routing on interface serial 0
clns enable
```

Command Summary

This section lists the commands described in this chapter in alphabetical order.

[**no**] **clns enable**

Enables OSI routing on an interface if the interface is not already configured for IS-IS or ISO-IGRP routing.

[**no**] **clns routing**

Configures a router for routing OSI protocols

[**no**] **decnet cost** *cost*

Sets a routing cost value for an interface and enables DECnet routing for the interface.

[**no**] **decnet routing** *decnet-address*

Enables DECnet routing on an interface.

encapsulation ppp

Enables PPP encapsulation on an interface.

A System Error Messages

Use this appendix with the Release 9.1 *System Error Messages* publication for a complete list of Release 9.14 error messages.

Introduction

This appendix lists and describes new system error messages for Release 9.14., which were not available at the time that the *System Error Messages* publication was printed. The system software sends these error messages to the console (and optionally, to a logging server on another system) during operation.

Not all of these messages indicate problems with your system. Some are informational, and others may help to diagnose problems with communication lines, internal hardware, or the system software.

MOP Error Messages

This section describes the error messages related to Maintenance Operation Protocol (MOP).

Error Message:

`Filename exceeds 16 characters`

Explanation:

Filename exceeds 16 characters. .

Recommended Action:

Rename the file so that the name contains 16 characters or fewer.

Error Message:

`MOP not enabled`

Explanation:

MOP is not enabled on an interface..

Recommended Action:

Enable MOP using the MOP ENABLED command.

Error Message:

```
MOP: Timed out on load assistant solicit
```

Explanation:

A MOP boot server did not respond to a "program request message" (a message soliciting the file to be copied into flash).

Recommended Action:

1. Check to make sure the filename is correct.
2. Check to make sure the MOP server is not down or disconnected from the network or not configured properly for MOP
3. Check to make sure MOP is enabled on the desired interface in your router configuration.
4. Check possible paths to the MOP server(s). A link may be down or intermediate routers/ bridges may not be properly configured for MOP.

Error Message:

```
MOP [interface type] [interface number]: Loading from [MAC-address] timed out
```

Explanation:

Timeout after the default or configured number retries, where [interface type] [interface number] could be "Serial0" or "Ethernet0." The message you have sent has not been received.

Recommended Action:

1. Try increasing the maximum number of retries through the MOP RETRIES *count* configuration command.
2. Check the MOP server and/or possible paths to the MOP server. The server or a link may have gone down.

Error Message:

```
MOP: Buffer overflow: [data-length]/[buffer-length]
```

Explanation:

There is not enough memory to netboot the image.

Recommended Action:

Your router needs more memory. Call your technical support representative.

Error Message:

```
Unrecognized MOP keyword - [offending word]
```

Explanation:

Syntax error in any MOP configuration command.

Recommended Action:

Check to make sure your command syntax is correct.

Error Message:

```
Unrecognized device code - [offending word]
```

Explanation:

Device code is not "ds200" or "cisco" in the MOP DEVICE-CODE *device* configuration command.

Recommended Action:

Make sure you are using the correct device code in the MOP DEVICE CODE command.

Error Message:

```
No MOP device code specified
```

Explanation:

No device code specified in the MOP DEVICE-CODE configuration command.

Recommended Action:

Make sure you have specified and device code in the MOP DEVICE-CODE *device* command.

Error Message:

```
MOP is only allowed on ethernet and serial interfaces
```

Explanation:

An unacceptable interface is specified in the MOP ENABLED command.

Recommended Action:

Check interface selected through the INTERFACE *type unit* command. The type must be an interface that supports MOP: serial or Ethernet.

Error Message:

`MOP enabled command requires an interface`

Explanation:

The MOP ENABLED command is specified without an interface name.

Recommended Action:

Make sure you use the INTERFACE *type unit* command to specify an interface before you type the MOP ENABLED command.

Error Message:

`Retransmit timer must be in range 1 - 20`

Explanation:

Retransmit timer value is not within the acceptable range.

Recommended Action:

Make sure that you are using a value between 1 and 20 for the argument *seconds* in the MOP RETRANSMIT-TIMER *seconds* configuration command.

Error Message:

`No MOP retransmit timer specified`

Explanation:

A retransmit timer value is not specified in the MOP RETRANSMIT-TIMER *seconds* configuration command.

Recommended Action:

Make sure you use a value between 1 and 20 for the argument *seconds* .

Error Message:

`Retry count must be in range 3 - 24`

Explanation:

Retry count value is not within the acceptable range in the MOP RETRIES *count* configuration command.

Recommended Action:

Make sure you use a value between 3 and 24 for the argument *count*.

Error Message:

```
No MOP retry maximum specified
```

Explanation:

Retry count value is not specified in the MOP RETRY *count* configuration command.

Recommended Action:

Make sure you specify a value between 3 and 24 for the argument *count* in the MOP RETRY *count* configuration command.

Error Message:

```
MOP sysid command requires an interface
```

Explanation:

An interface is not specified for the MOP SYSID configuration command.

Recommended Action:

Make sure you enter an INTERFACE *type unit* command to specify an interface before you type the MOP SYSID command.

Error Message:

```
MOP argument missing
```

Explanation:

No argument is specified with a MOP configuration command.

Recommended Action:

Make sure your command syntax includes the required arguments.

B
Access List Summary

This appendix summarizes the general command syntax and number ranges (or symbolic names) used for the access lists supported by the DECbrouter 90 software. The summaries are listed by protocol, in alphabetical order. The command to create the access list is given first, followed by the command you use to assign the access list.

Access list ranges are included in the summary descriptions; however, in actual use, only one number is selected from the given range. Table B–1 (at the end of this appendix) lists the access list number ranges in numerical order.

Note

This summary provides a general listing of the DECbrouter 90 access list forms. However, this summary is not complete. For a more detailed discussion of using access lists, refer to the specific chapters for discussions of associated filtering mechanisms.

Apollo Domain Access List

Access list specifications:

apollo access-list *name* {**permit** | **deny**} [*firstnet*-] *lastnet.host* [*wildcard-mask*]

Interface assignment command:

apollo access-group *name*

Access List Summary

AppleTalk Access List

Access list specifications:

access-list *600-699* {**permit** | **deny**} **network** *network*

access-list *600-699* {**permit** | **deny**} **cable-range** *start-end*

access-list *600-699* {**permit** | **deny**} **includes** *start-end*

access-list *600-699* {**permit** | **deny**} **within** *start-end*

access-list *600-699* {**permit** | **deny**} **zone** *zonename*

access-list *600-699* {**permit** | **deny**} **additional-zones**

access-list *600-699* {**permit** | **deny**} **other-access**

Interface assignment commands can be one of the following:

appletalk access-group *600-699*

appletalk distribute-list *600-699* **in**

appletalk distribute-list *600-699* **out**

appletalk getzonelist-filter *600-699*

DECnet Access List

Access list specification can be one of the following:

access-list *300-399* {**permit** | **deny**} *destination destination-mask*

access-list *list* {**permit** | **deny**} *source source-mask destination destination-mask*

access-list *300-399* {**permit** | **deny**} *source source-mask* [*destination destination-mask*] [*connect-entries*]

Interface assignment commands can be one of the following:

decnet access-group *300-399*

decnet in-routing filter *300-399*

decnet out-routing-filter *300-399*

Ethernet Address Access List

Access list specifications:

access-list *700-799* {**permit** | **deny**} *address mask*

Interface assignment command:

bridge-group *1-9* {**input-address-list** | **output-address-list**} *700-799*

Ethernet Protocol Access List

Access list specifications:

access-list *200-2990* {**permit** | **deny**} *0xtype-code 0xmask*

Interface assignment command:

bridge-group *1-9* {**input-type-list** | **output-type-list**} *200-299*

IP Access Lists

The following variations of IP access lists are available.

Standard IP Access List

Access list specifications:

access-list *1-99* {**permit** | **deny**} *address mask*

Interface/line assignment commands can be one of the following:

ip access-group *1-99*

access-class *1-99* {**out** | **in**} (for terminal line assignment)

Router configuration command assignment:

distance weight [*address mask*] [*1-99*]

distribute-list *1-99* **in** [*interface-name*]

distribute-list *1-99* **out** [*interface-name* | *routing-process*]

offset-list *1-99* {**in** | **out**} *offset* (add an offset to metrics for networks)

Access List Summary

BGP Access Lists

Access list specification:

ip as-path access-list *1-99* [**permit** | **deny**] *as-regular-expression*

Router assignment command:

neighbor *address* **distribute-list** *1-99* (for BGP filtering BGP advertisements)

neighbor *address* **filter-list** *1-99* {**in** | **out** | **weight** *weight*}

EGP Access Lists

Access list specifications:

access-list *1-99* {**permit** | **deny**} *address mask*

Router assignment command:

neighbor any [*1-99*]

SLIP Access Lists

Access list specifications:

access-list *1-99* {**permit** | **deny**} *address mask*

Interface assignment command:

slip access-class *1-99* {**in** | **out**}

Extended IP Access List

Access list specifications:

access-list *100-199* {**permit** | **deny ip** | **tcp** | **udp** | **icmp**} *source source-mask dest dest-mask* [**lt** | **gt** | **eq** | **neq** *dest-port*]

Interface assignment command:

ip access-group *100-199*

Novell Access Lists

Standard Novell Access Lists

Access list specifications:

access-list *800-899* {**deny** | **permit**} *novell-source-network*[[.*source-address* [**source-mask**]] *novell-destination-network* [*destination-address* [*destination-mask*]]

Interface assignment command:

novell access-group *800-899*

Extended Novell Access Lists

Access list specifications:

access-list *900-999* {**deny** | **permit**} *novell-protocol source-network.* [*source-address* [*source-mask*]] *source-socket destination-network.* [*destination-address* [*destination-mask*]] *destination-socket*

Interface assignment command:

novell access-group *900-999*

Novell SAP Access List Filter

Access list specifications:

access-list *1000-1099* {**permit** | **deny**} *network*.[*address*] [*service-type*]

Global configuration assignment commands:

novell input-sap-filter *1000-1099*

novell output-sap-filter *1000-1099*

novell router-sap-filter *1000-1099*

Source-Route Bridge Protocol Type Access List

Access list specifications:

access-list *200-299* {**permit** | **deny**} *type-code wild-mask*

netbios access-list bytes *name* {**permit** | **deny**} *offset pattern*

netbios access-list host *name* {**permit** | **deny**} *pattern*

Transparent Bridge Access List

Access list specifications:

access-list *200-2990* {**permit** | **deny**} *type-code wild-mask*

access-list *700-799* {**permit** | **deny**} *address-mask*

Interface assignment command:

bridge-group *200-299* **input-address-list** *list*

VINES Access List

Access list specification can be one of the following:

vines access-list *1-100* {**permit** | **deny**} **IP** *source-address source-mask dest-address dest-mask*

vines access-list *1-100* {**permit** | **deny**} *protocol source-address source-mask source-port dest-address dest-mask dest-port*

Interface assignment command:

vines access-group *list*

XNS Access Lists

Access list specifications:

access-list *400-499* {**permit** | **deny**} *net* [*source-address*] [*source-mask*] *net* [*dest-address*] [*dest-mask*]

Interface assignment command:

xns access-group *400-499*

Extended XNS Access Lists

Access list specifications:

> **access-list** *500-599* {**permit** | **deny**} *xns-protocol net* [*source-address*] [*source-mask*] *source-socket net* [*dest-address*] [*dest-mask*] *dest-socket*

Interface assignment command:

> **xns access-group** *500-599*

Table B–1 Summary of Numerical Ranges

Protocol	Range
IP	1—99
Extended IP	100—199
Ethernet type code	200—299
DECnet	300—399
XNS	400—499
Extended XNS	500—599
AppleTalk	600—699
Ethernet address	700—799
Novell	800—899
Extended Novell	900—999
Novell SAP	1000—1099

C
Ethernet Type Codes

This appendix lists known Ethernet type codes for use with type code filtering configuration commands.

Note

Not all codes are used on Ethernet media.

Hexadecimal	Description (Notes)
0000-05DC	IEEE 802.3 Length Field
0101-01FF	Experimental; for development (Conflicts with 802.3 length fields)
0200	Xerox PUP (Conflicts with IEEE 802.3 length fields)
0201	Xerox PUP Address Translation (Conflicts with IEEE 802.3 length fields)
0600	Xerox XNS IDP
0800	DOD Internet Protocol (IP) *[1]#[2]
0801	X.75 Internet
0802	NBS Internet
0803	ECMA Internet
0804	CHAOSnet
0805	X.25 Level 3
0806	Address Resolution Protocol (for IP and CHAOS)
0807	XNS Compatibility
081C	Symbolics Private
0888-088A	Xyplex
0900	Ungermann-Bass Network Debugger
0A00	Xerox IEEE 802.3 PUP
0A01	Xerox IEEE 802.3 PUP Address Translation
0BAD	Banyan VINES IP
0BAE	Banyan VINES Loopback
0BAF	Banyan VINES Echo
1000	Berkeley trailer negotiation

[1]An asterisk (*) indicates the current correction in various informational displays.

[2]A pound sign (#) is a delimiting character for configuration commands that contain arbitrary text string.

Ethernet Type Codes

Hexadecimal	Description (Notes)
1001-100F	Berkeley trailer encapsulation for IP
1600	VALID system protocol
4242	PCS Basic Basic Block Protocol
5208	BBN Simnet Private
6000	DEC unassigned
6001	DEC Maintenance Operation Protocol (MOP) Dump/Load Assistance
6002	DEC MOP Remote Console
6003	DEC DECNet Phase IV
6004	DEC Local Area Transport (LAT)
6005	DEC DECNet Diagnostics
6006	DEC DECNet Customer Use
6007	DEC Local Area VAX Cluster (LAVC)
6008	DEC unassigned
6009	DEC unassigned
6010-6014	3Com Corporation
7000	Ungermann-Bass (UB) Download
7001	{item}UB diagnostic/loopback
7002	UB diagnostic/loopback
7020-7029	LRT (England)
7030	Proteon
8003	Cronus VLN
8004	Cronus Direct
8005	HP Probe protocol
8006	Nestar
8008	AT&T
8010	Excelan
8013	Silicon Graphics diagnostic (obsolete)
8014	Silicon Graphics network games (obsolete)
8015	Silicon Graphics reserved type (obsolete)
8016	Silicon Graphics XNS NameServer, bounce server (obsolete)
8019	Apollo Domain
802E	Tymshare
802F	Tigan, Inc.
8035	Reverse Address Resolution Protocol (RARP)
8036	Aeonic Systems
8038	DEC LANBridge Management
8039	DEC unassigned
803A	DEC unassigned
803B	DEC unassigned
803C	DEC unassigned

Hexadecimal	Description (Notes)
803D	DEC Ethernet CSMA/CD Encryption Protocol
803E	DEC unassigned
803F	DEC LAN Traffic Monitor Protocol
8040	DEC unassigned
8041	DEC unassigned
8042	DEC unassigned
8044	Planning Research Corp.
8046-8047	AT&T
8049	ExperData (France)
805B	Versatile Message Translation Protocol RFC-1045 (Stanford)
805C	Stanford V Kernel, production
805D	Evans & Sutherland
8060	Little Machines
8062	Counterpoint Computers
8065-8066	University of Mass. at Amherst
8067	Veeco Integrated Automation
8068	General Dynamics
8069	AT&T
806A	Autophon (Switzerland)
806C	ComDesign
806D	Compugraphic Corporation
806E-8077	Landmark Graphics Corporation
807A	Matra (France)
807C	Merit Internodal
807D-8080	Vitalink Communications
8080	Vitalink TransLAN III Management
8081-8083	Counterpoint Computers
8088-808A	Xyplex
809B	Kinetics EtherTalk (AppleTalk over Ethernet)
809C-809E	Datability
809F	Spider Systems Ltd.
80A3	Nixdorf Computers (West Germany)
80A4-80B3	Siemens Gammasonics Inc.
80C0-80C3	Digital Communications Assoc. Inc.
80C1	DCA Data Exchange Cluster
80C4	Banyan VINES IP
80C5	Banyan VINES Echo
80C6	Pacer Software
80C7	Applitek Corporation
80C8-80CC	Intergraph Corporation
80CD-80CE	Harris Corporation

Ethernet Type Codes

Hexadecimal	Description (Notes)
80CF-80D2	Taylor Instrument
80D3-80D4	Rosemount Corporation
80D5	IBM SNA Services over Ethernet
80DD	Varian Associates
80DE	Integrated Solutions Transparent Remote File System (TRFS)
80DF	Integrated Solutions
80E0-80E3	Allen-Bradley
80E4-80F0	Datability
80F2	Retix
80F3	Kinetics AppleTalk Address Resolution Protocol (AARP)
80F4-80F5	Kinetics
80F7	Apollo Computer
80FF-8103	Wellfleet Communications
8069	AT&T
807B	Dansk Data Elektronik A/S
8107	Symbolics Private
8108	Symbolics Private
8109	Symbolics Private
8130	Waterloo Microsystems Inc.
8131	VG Laboratory Systems
8137	Novell NetWare IPX (old)
8137-8138	Novell, Inc.
8139-813D	KTI
9000	Loopback (Configuration Test Protocol)
9001	Bridge Communications XNS Systems Management
9002	Bridge Communications TCP/IP Systems Management
FF00	BBN VITAL LANBridge cache wakeups

D
Pattern Matching

This appendix describes the pattern-matching scheme used by the X.25 switching feature.

Regular Expression Pattern Matching

The X.121 address and Call User Data are used to find a matching routing table entry. The list is scanned from the beginning to the end and each entry is pattern-matched with the incoming X.121 address and Call User Data to the X.121 and Call User Data in the routing table entry. If the pattern match for both entries succeeds, then that route is used. If the incoming call does not have any Call User Data, then only the X.121 address pattern match need succeed with an entry that only contains an X.121 pattern. If Call User Data is present, and while scanning, a route is found that matches the X.121 address but does not have a Call User Data pattern, then that route is used when an exact match cannot be found.

Regular expressions are used to allow pattern-matching operations on the X.121 addresses and Call User Data. The most common operation is to do prefix matching on the X.121 DNIC field and route accordingly. For example, the pattern ^3306 will match all X.121 addresses with a DNIC of 3306. The caret (^) is a special regular expression character that means to anchor the match at the beginning of the pattern.

If a matching route is found, the incoming call is forwarded to the *next hop* depending on the routing entry. If no match is found, the call is cleared. If the route specifies a serial interface running X.25, the call will attempt to be forwarded over that interface. If the interface is not operational or out of available virtual circuits, the call will be cleared. Otherwise, the expected Clear Request or Call Accepted message will be forwarded back toward the originator. The "null 0" interface can be used to early-terminate or refuse calls to specific locations.

If the matching route specifies an IP address, a TCP connection will be established to port 1998 at the specified IP address, which must be another DECbrouter 90 or Cisco router. The Call Request packet will be forwarded to the remote router, where it will be processed in a similar fashion. If a routing table entry is not present or the serial interface is down or out of virtual circuits, a Clear Request will be sent back and the TCP connection will be closed. Otherwise, the call will be forwarded over the serial interface and the expected Clear Request or Call Accepted packet will be returned. Incoming calls received via TCP connections that match a routing entry specifying an IP address will be cleared. This restriction prevents Digital routers from establishing a TCP connection to another router that would establish yet another TCP connection. A router must always connect to the remote router with the destination DTE directly attached.

Regular expressions provide a way to specify wide ranges of X.121 addresses and Call User Data fields by using just a few keystrokes. If you are familiar with regular expressions from UNIX programs such as *regexp*, you are already familiar with much of Digital Systems' regular expression implementation.

Writing regular expressions is simple once you see and try a few examples. A regular expression is a formula for generating a set of strings. If a particular string can be generated by a given regular expression, then that string and regular expression *match*. In many ways, a regular expression is a program, and the regular expression matches the strings it generates.

A regular expression is built up of different components, each of which is used to build the regular expression string-generating program. The simplest usable component is the *atom*, but first you need to understand *ranges*, because atoms are built of these.

Ranges

A range is a sequence of characters contained within left and right square brackets ([ ]). A character matches a range if that character is contained within the range; for example, the following syntax forms the range consisting of the characters "a," "q," "c," "s," "b," "v," and "d." The order of characters is usually not important; however, there are exceptions and these will be noted.

[aqcsbvd]

You can specify an ASCII sequence of characters by specifying the first and last characters in that sequence, and separating them with a hyphen (-).

[a-dqsv]

The above example could also be written so as to specify right square brackets ( ]) as a character in a range. To do so, enter the bracket as the *first* character after the initial left square bracket that starts the range.

This example matches right bracket and the letter "d."

[ ]d]

To include a hyphen (-), enter it as either the first or the last character of the range.

You can reverse the matching of the range by including a wedge (caret) at the start of the range. This example matches any letter *except* the ones listed. When using the wedge with the special rules for including a bracket or hyphen, make the wedge the very first character.

[^a-dqsv]

This example matches anything except a right square bracket ( ])or the letter "d":

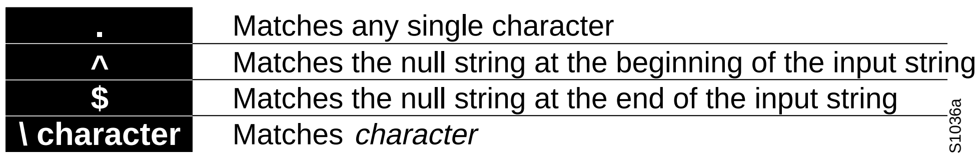
[^]d]

Atoms

Atoms are the most primitive usable part of regular expressions. An atom can be as simple as a single character. The letter "a" is an atom, for example. It is also a very simple regular expression, that is, a program that generates only one string, which is the single-letter string made up of the letter "a." While this may seem trivial, it is important to understand the set of strings that your regular expression program generates. As will be seen in upcoming explanations and examples, much larger sets of strings can be generated from more complex regular expressions.

Certain characters have a special meaning when used as atoms; refer to Table D–1.

Table D–1 Special Symbols Used as Atoms



.	Matches any single character
^	Matches the null string at the beginning of the input string
$	Matches the null string at the end of the input string
\ character	Matches *character*

Note another use for the ^ symbol.

As an example, the regular expression matches "abcd" only if "abcd" starts the full string to be matched:

```
^abcd
```

Whereas the following expression is an atom that is a range that matches any single letter, as long as it is not the letters "a," "b," "c," or "d."

```
[^abcd]
```

It was previously stated that a single character string such as the letter "a" is an atom. A character by itself, such as $, means "match the null string at the end of the input string."

```
$
```

Whereas this atom matches a dollar sign ($). Preceding a character with a backslash (\) removes the special meaning of that character.

```
\$
```

Any character can be preceded with the backslash character with no adverse affect.

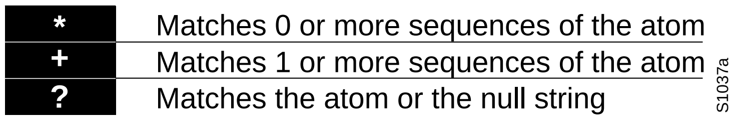
```
\a
```

Atoms are also full regular expressions surrounded by parentheses. For example, both "a" and "(a)" are atoms matching the letter "a." This will be important later, as we see patterns to manipulate entire regular expressions.

Pieces

A piece is an atom optionally followed by one of the symbols listed in Table D–2.

Table D–2 Special Symbols Used with Pieces



*	Matches 0 or more sequences of the atom
+	Matches 1 or more sequences of the atom
?	Matches the atom or the null string

This example matches any number of occurrences of the letter "a," including *none*.

a*

This string requires there to be at least one letter "a" in the string to be matched:

a+

This string means that the letter "a" can be there *once*, but it does not have to be:

a?

This string matches any number of asterisks (*).

\ **

Here is an example using parentheses. This string matches any number of the two-atom string "ab."

(ab)*

As a more complex example, this string matches one or more instances of alphanumeric pairs (but not none, that is, an *empty string* is not a match):

([A-Za-z][0-9])+

The order for matches using the optional *, +, or ? symbols is longest construct first. Nested constructs are matched from outside to inside. Concatenated constructs are matched beginning at the left side of the construct.

Branch

A branch is simply a set of zero or more concatenated pieces. The previous alphanumeric example was an example of a branch as concatenated pieces. Branches are matched in the order normally read—from left to right. For example, in the previous example, the regular expression matches "A9b3," but not "9Ab3" because the alphabet is given first in the two-atom branch of [A-Za-z][0-9].

Regular Expressions

A regular expression is a branch or any number of branches separated by a vertical bar (|). A string is said to match the regular expression if it is generated by the "program" specified in any of the branches. Of course, a string can be generated by more than one branch. For example, "abc" is generated by all branches in the regular expression

abc | a*bc+ | ab?c | .*

Also remember that if a regular expression can match two different parts of an input string, it will match the earliest part first.

The regular expression support and the technical information for this portion of the documentation is based on Henry Spencer's public domain *rgrep(3)* library package.

E
ASCII Character Set

Many of the configuration commands described in this manual require the decimal representation of an ASCII character. Figure E–1 provides translations from ASCII character to decimal number. To find the decimal equivalent of the ASCII character, add the row heading and column heading numbers together. To find the ASCII code for the percent (%) character, for example, add row 30 to column 7, so the ASCII code is 37.

Figure E–1 ASCII-to-Decimal Translation Table

	0	1	2	3	4	5	6	7	8	9
0	^@	^A	^B	^C	^D	^E	^F	^G	^H	^I
10	^J	^K	^L	^M	^N	^O	^P	^Q	^R	^S
20	^T	^U	^V	^W	^X	^Y	^Z	ESC	FS	GS
30	RS	US	SP	!	"	#	$	%	&	'
40	(	)	*	+	,	_	.	/	0	1
50	2	3	4	5	6	7	8	9	:	;
60	<	=	>	?	@	A	B	C	D	E
70	F	G	H	I	J	K	L	M	N	O
80	P	Q	R	S	T	U	V	W	X	Y
90	Z	[	\	]	^	_	'	a	b	c
100	d	e	f	g	h	i	j	k	l	m
110	n	o	p	q	r	s	t	u	v	w
120	x	y	z	{	|	{	~	DEL		

1320

F

References and Recommended Reading

Presented here are lists of references used in this publication and suggested references for understanding concepts presented here. Some tips for obtaining these references also are provided following the lists.

Commercially Available Publications

Black, Uyless. *Physical Level Interfaces and Protocols*, IEEE Computer Society Press, 1988.

Black, Uyless. X.25 *and Related Protocols*, IEEE Computer Society Press, 1991.

Comer, Douglas E. *Internetworking with TCP/IP: Principles, Protocols, and Architecture*. Vol. I, Prentice-Hall, 1988.

Davidson, John. *An Introduction to TCP/IP*. Springer-Verlag, 1988.

Martin James and the ARBEN Group. *Local Area Networks*. Prentice Hall, 1989.

McNamara, John E. *Local Area Networks*. Digital Press, Educational Services, Digital Equipment Corporation, 12 Crosby Drive, Bedford, Massachusetts 01730.

Meijer, Anton. *Systems Network Architecture: A Tutorial*. John Wiley & Sons, Inc., 1987.

Miller, Mark A. *LAN Protocol Handbook*. M&T Publishing, 1990.

Rose, Marshall T. *The Simple Book: An Introduction to Management of TCP /IP-based Internets*. Prentice Hall, 1991.

Schlar, Sherman K. *Inside X.25: A Manager's Guide*. McGraw-Hill, Inc., 1990.

Sherman, Ken. *Data Communications: A User's Guide*, 2nd edition, Reston Publishing Company, Inc., 1985.

Sidhu, Gursharan S., Richard F. Andrew, and Alan Oppenheimer. *Inside AppleTalk*, Second Edition, Addison-Wesley Publishing Company, 1990

Stallings, William. *Handbook of Computer Communications Standards*, Vols. 1-3, Macmillan Publishing Company, 1987.

Technical Publications and Standards

ANSI X3T9.5 Committee. *FDDI Station Management* (SMT). Rev. 6.1, March 15, 1990.

Apple Computers. *AppleTalk Network System Overview*. Addison-Wesley Publishing Company, Inc., 1989.

Bellcore. *Generic System Requirements in Support of a Switched Multi-Megabit Data Service*. Technical Advisory, TA-TSY-000772, October, 1989.

——. *Local Access System Generic Requirements, Objectives, and Interface Support of Switched Multi-Megabit Data Service*. Technical Advisory TA-TSY-000773, Issue 1, December, 1985.

——.*Switched Multi-megabit Data Service (SMDS) Operations Technology Network Element Generic Requirements*, Technical Advisory TA-TSY-000774

CCITT. *CCITT Data Communications Networks - Services and Facilities, Terminal Equipment and Interfaces, Recommendations X.1-X.29*, Yellow Book, Volume VIII, Fascicle VIII.2, 1984.

CCITT. *CCITT Data Communications Networks - Interfaces, Recommendations X.20-X.32, Red Book*, Volume VIII, Fascicle VIII.3, 1984.

Cisco Systems. *HSSI Specifications*.

Cisco Systems. *SDLLC Application Note*, 1991.

Digital Equipment Corporation. *DECNET/OSI Phase V: Making the Transition From Phase IV*. EK-PVTRN-BR, 1989.

——. *DECserver 200 Local Area Transport (LAT) Network Concepts*. AA-LD84A-TK, June 1988.

——. *DIGITAL Network Architecture (Phase V)*, EK-DNAPV-GD-001, September 1987.

Digital Equipment Corporation, Intel Corporation, Xerox Corporation. *The Ethernet: A Local Area Network, Data Link Layer and Physical Layer Specifications, digital, intel, XEROX*. Version 2.0, November 1982.

Hemrick, C. and L. Lang, "Introduction to Switched Multi-megabit Data Service (SMDS), an Early Broadband Service," publication pending in the Proceedings of the XIII International Switching Symposium (ISS 90), May 27, 1990-June 1, 1990.

Hewlett-Packard. X.25: *The PSN Connection; An Explanation of Recommendation X.25*, 5958-3402, October 1985.

IBM. *Advanced Communications Function for VTAM, general information: concepts*. GS27-0463.

——.*Advanced Communications Function for VTAM (ACF/VTAM), general information: introduction*. GS27-0462.

——. *ACF/NCP/VS network control program, system support programs: general information*. GC30-3058.

——. *Dictionary of Computing*, SC20-1699-7, 1987.

——. *Local Area Network Technical Reference.*

——. *Network Communications Control Facility: general information*, GC27-0429.

——. *Network Problem Determination Application: general information*, GC34-2010.

——. *Synchronous Data Link Control: general information*, GA27-3093.

——. *Systems Network Architecture-Architecture Logic*, SC30-3112.

——. *Systems Network Architecture: concepts and products*, GC30-3072.

——. *Systems Network Architecture format and protocol reference manual: architectural logic*, SC30-3112, November 1980.

——. *Systems Network Architecture - introduction to sessions between Logical Units*, GC20-1869.

——. *Systems Network Architecture - Logical Unit types*, GC20-1868.

——. *Systems Network Architecture: reference summary*, GA27-3136.

——. *Systems Network Architecture-Session between Logical Units*, GC20-1868.

——. *Systems Network Architecture: technical overview*, GC30-3073-1, 1985.

——. *Systems Network Architecture Transaction Programmer's Reference Manual for LU Type 6.2.*

——. *Token-Ring Network Architecture Reference.*

——. *The X.25 1984 Interface for Attaching IBM SNA Nodes to Packet-Switched Data Networks: Architecture Reference.*

——. *The X.25 1984 Interface for Attaching IBM SNA Nodes to Packet-Switched Data Networks: General Information Manual.*

IEEE 802.2 - *Local Area Networks Standard, 802.2 Logical Link Control*, ANSI /IEEE Standard, October 1985.

IEEE 802.3 - *Local Area Networks Standard, 802.3 Carrier Sense Multiple Access*, ANSI/IEEE Standard, October, 1985. IEEE Project 802 - *Local & Metropolitan Area Networks. Proposed Standard: Distributed Queue Dual Bus (DQDB) Subnetwork of a Metropolitan Area Network (MAN)*, February 7, 1990.

McGraw-Hill/Data Communications. *McGraw-Hill's Compilation of Data Communications Standards*, Edition III, 1986.

National Security Agency. *Blacker Interface Control Document (ICD).*

StrataCom. *Frame Relay Interface Specification*. 040-207460, Rev. 2.3, August 9, 1990.

XEROX. *Internet Transport Protocols*. XNSS 029101, January 1991.

Obtaining Technical Information

Many of the references cited in this list can be obtained from book stores or ordered directly from the publisher or company that produced the reference. In the world of data communications, however, there are numerous standards that are kept by various companies and agencies. This section provides some tips on obtaining this type information.

Obtaining RFCs

Information about the Internet suite of protocols is contained in documents called *Requests for Comments* or RFCs. These documents are maintained by Government Systems, Inc. (GSI). You can request copies by contacting GSI directly, or you can use the TCP/IP File Transfer Protocol (FTP) to obtain an electronic copy.

Contacting GSI

You can contact GSI through mail, by telephone, or through electronic mail.

Government Systems, Incorporated
Attn: Network Information Center
14200 Park Meadow Drive, Suite 200
Chantilly, Virginia 22021
1-800-365-3642
(703) 802-4535
(703) 802-8376 (FAX)

`NIC@NIC.DDN.MIL`

Network address: 192.112.36.5
Root domain server: 192.112.36.4

Obtaining Technical Standards

Following are some places from which you can obtain technical standards:

- Omnicom at 1-800-OMNICOM.
- Global Engineering Documents, 2805 McGraw Avenue, Irvine, California, 92714. Telephone: 1-800-854-7179.
- American National Standards Institute, 1430 Broadway, New York, New York, 10018. Telephone: 212-642-4932 and 212-302-1286.
- IEEE Computer Society Press, Customer Service Center, 10662 Los Vaqueros Circle, P.O. Box 3014, Los Alamitos, CA, 90720-1264. Telephone: 714-821-8380.

G
X.25 Diagnostic Codes

The two tables in this appendix contain information about X.25 diagnostic codes including:

- Differences between the DECbrouter 90 implementation and the CCITT definitions
- Diagnostic code numbers and descriptions

Table G–1 describes the differences between Digital's implementation of certain X.25 network-generated, "international problem" diagnostic fields and the definitions provided in Annex E of CCITT Recommendation X.25. The complete diagnostics listing is provided in Annex E's Table E-1/X.25.This translation table lists the decimal value of the diagnostic number, the CCITT description, and the Digital definition.

Table G–1 Annex E International Problem Diagnostic Code Differences

Decimal Value	Annex E, Rec. X.25 Diagnostic Description	Digital Proprietary Definition of Diagnostic Codes
112	International problem	Not used.
113	Remote network problem	Not used.
114	International protocol problem	Not used.
115	International link out of order	Indicates one of the following failures: failed when initializing a switched PVC; in TCP tunneling, failed when initiating or resetting a PVC; or, failed when PAD PVC circuit is being initiated or reset.
116	International link busy	Not used.
117	Transit network facility problem	Not used.
118	Remote network facility problem	Not used.
119	International routing problem	Indicates the following failure: in TCP tunneling of X.25 when session is closed by network.
120	Temporary routing problem	Indicates the following failure: when tunneling X.25 through TCP/IP and the remote network is identified as unreachable.
121	Unknown called DNIC	Not used.

(continued on next page)

Table G–1 (Cont.) Annex E International Problem Diagnostic Code Differences

Decimal Value	Annex E, Rec. X.25 Diagnostic Description	Digital Proprietary Definition of Diagnostic Codes
122	Maintenance action (may apply to maintenance action within a national network)	For CMNS, indicates the following: router fails to route the call due to setup or unreachability of destination; when VC is cleared using the CLEAR X25-VC EXEC command.

The following table is summarized from the 1988 CCITT Recommendation X.25 Tables 20/X.25, 21/X.25, 22/X.25, 23/X.25, and E-1/X.25. The International Telecommunication Union holds the copyright to the CCITT Recommendations as published in the CCITT Blue Book Volume VIII—Fascicle VIII.2, "Data Communication Networks: Services and Facilities, Interfaces. Recommendations X.1-X.32," Geneva 1989, ISBN 92-61-03671-6.

This table allows users to easily interpret the cause and diagnostic fields reported in various packets. Digital equipment does not generate all of the codes listed, nor is the protocol affected by the cause or diagnostic code reported in a packet. Note that Digital equipment reports cause and diagnostic fields in their decimal representations.

Table G–2 X.25 Cause and Diagnostic Field Descriptions

Cause Description	Decimal Value	Hex Value
1. Cause field codes for Restart packets		
Local procedure error	1	01
Network congestion	3	03
Network operational	7	07
Registration/cancellation confirmed	127	7F
2. Cause field codes for Clear packets		
DTE originated	0,	00,
	128-255	80-FF
Number busy	1	01
Invalid facility request	3	03
Network congestion	5	05
Out of order	9	09
Access barred	11	08
Not obtainable	13	1D
Remote procedure error	17	11
Local procedure error	19	13
RPOA out of order	21	15
Reverse charging acceptance not subscribed	25	19

(continued on next page)

Table G–2 (Cont.) X.25 Cause and Diagnostic Field Descriptions

Cause Description	Decimal Value	Hex Value
Incompatible destination	33	21
Fast select acceptance not subscribed	41	29
Ship absent	57	39
3. Cause field codes for reset packets		
DTE originated	0,	00,
	128-255	80-FF
Out of order	1	01
Remote procedure error	3	03
Local procedure error	5	05
Network congestion	7	07
Remote DTE operational	9	09
Network operational	15	0F
Incompatible destination	17	11
Network out of order	29	1D
4. Cause field codes for Registration		
Confirmation packets		
Invalid facility request	3	03
Network congestion	5	05
Local procedure error	19	13
Registration/cancellation confirmed	127	7F

Diagnostic Description	Decimal Value	Hex Value
5. Diagnostic field codes for all packet types		
No additional information	0	00
Invalid P (S)	1	01
Invalid P (R)	2	02
Packet type invalid	16	10
For state r1	17	11
For state r2	18	12
For state r3	19	13
For state p1	20	14
For state p2	21	15
For state p3	22	16
For state p4	23	17
For state p5	24	18

X.25 Diagnostic Codes

Diagnostic Description	Decimal Value	Hex Value
For state p6	25	19
For state p7	26	1A
For state d1	27	1B
For state d2	28	1C
For state d3	29	1D
Packet not allowed	32	20
Unidentifiable packet	33	21
Call on one-way logical channel	34	22
Invalid packet type on a permanent virtual circuit	35	23
Packet on unassigned logical channel	36	24
Reject not subscribed to	37	25
Packet too short	38	26
Packet too long	39	27
Invalid general format identifier	40	28
Restart or registration packet with nonzero in bits 1 to 4 of octet 1, or bits 1 to 8 of octet 2	41	29
Packet type not compatible with facility	42	2A
Unauthorized interrupt confirmation	43	2B
Unauthorized interrupt	44	2C
Unauthorized reject	45	2D
Time expired	48	30
For incoming call	49	31
For clear indication	50	32
For reset indication	51	33
For restart indication	52	34
For call deflection	53	35
Call setup, call clearing, or registration problem	64	40
Facility/registration code not allowed	65	41
Facility parameter not allowed	66	42
Invalid called DTE address	67	43
Invalid calling DTE address	68	44
Invalid facility/registration length	69	45
Incoming call barred	70	46
No logical channel available	71	47
Call collision	72	48
Duplicate facility requested	73	49
Nonzero address length	74	4A
Nonzero facility length	75	4B
Facility not provided when expected	76	4C
Invalid CCITT-specified DTE facility	77	4D

Diagnostic Description	Decimal Value	Hex Value
Maximum number of call redirections or call deflections exceeded	78	4E
Miscellaneous	80	50
Improper cause code from DTE	81	51
Not aligned octet	81	52
Inconsistent Q bit setting	82	53
NIU problem	83	54
International problem	112	70
Remote network problem	113	71
International protocol problem	114	72
International link out of order	115	73
International link busy	116	74
Transit network facility	117	75
Remote network facility problem	118	76
International routing problem	119	77
Temporary routing problem	120	78
Unknown called DNIC	121	79
Maintenance action (may apply to maintenance action within a national network)	122	7A
CCITT—Reserved for network-specific diagnostic information	128-255	80-FF
ISO-DTE specific signals	160	A0
DTE operational	161	A1
DTE not operational	162	A2
DTE resource constraint	163	A3
Fast select not subscribed	164	A4
Invalid partially full data packet	165	A5
D-bit procedure not supported	166	A6
Registration/cancellation confirmed	167	A7

Index