SunNet X.25
System Administration Manual

Sun Microsystems, Inc. • 2550 Garcia Avenue • Mountain View, CA 94043 • 415-960-1300

Part No: 800-5248-06
pre-FCS version of 9 January 1991
SunNet 7.0

This software and documentation is based in part on the Fourth Berkeley Software Distribution under license from the Regents of the University of California. We acknowledge the following individuals and institutions for their role in its development: The Regents of the University of California, the Electrical Engineering and Computer Sciences Department at the Berkeley Campus of the University of California, and Other Contributors.

Preface

Purpose and Audience

This manual enables a system administrator of Sun™ workstations to configure and manage a SunNet™ X.25 gateway. This guide is a supplement to the SunOS™ manual *System and Network Administration*. This SunNet manual is aimed at system administrators who are very familiar with SunOS and who manage their installation's SunNet X.25 gateway. The intended reader should also be familiar with the CCITT X.25 recommendation and the requirements of the Packet-Switched Data Network (PSDN) or LAN that he or she intends to use.

Summary of Contents

Chapter 1, "Overview," introduces the SunLink X.25 product, including a discussion of each of its primary applications. The chapter presents a "road map" to this manual, telling you which chapters to use, based on the applications you plan to use. The chapter also includes an overview of the X.25 recommendation.

Chapter 2, "Overview of DDN Support," presents an overview of SunNet X.25's support for DDN X.25 Standard Service, which is used for connecting TCP/IP networks with Class A or B network addresses to the DDN.

Chapter 3, "Preparation for Using X.25," explains what you need to know to install and configure SunNet X.25. The chapter concludes with an installation questionnaire that facilitates the configuration task.

Chapter 4, "Connecting TCP/IP Networks," discusses IP routing issues when SunNet X.25 is used to connect TCP/IP networks.

Chapter 5, "Software Installation," tells you how to extract the SunNet X.25 product from its delivery medium and explains how remote clients can mount SunNet X.25 software from an NFS server.

Chapter 6, "X.25 Configuration," explains the use of `install.x25`, the SunNet X.25 configuration script. This chapter contains configuration instructions for all "off-the-shelf" applications of SunNet X.25.

Chapter 7, "Starting SunNet X.25," explains the commands used in the SunNet X.25 startup file. It also tells you how to stop SunNet X.25.

Chapter 8, "Configuring and Running as an HDLC Service," explains how to run user programs directly over the HDLC layer in SunNet X.25.

Chapter 9, "X.25 Troubleshooting," presents steps you can use to troubleshoot SunNet X.25. It also explains the use of a number of utilities you can use monitor and diagnose problems in the various layers of the product.

Appendix A, “PSDN-specific Parameter Settings,” lists parameter settings required by major public and private PSDNs throughout the world.

Appendix B, “The Link Configuration File,” describes in detail the parameter settings for the three X.25 layers, as they are stored in a special configuration file. It also describes the `x25config` utility, which uses data from your configuration file to configure a link.

Appendix C, “Link Configuration File for DDN X.25 Standard Service,” lists the link configuration parameters used for DDN X.25 Standard Service. It also explains how an X.121 address is extracted from a DDN (IP) host address.

Appendix D, “`x25mgr` Configuration File,” lists the parameters used in the configuration file for `x25mgr`, a program that manages virtual circuits for many uses of SunNet X.25. It also describes the syntax for `x25mgr` and the syntax and use of the `x25admin` program, also used for managing virtual circuits.

Appendix E, “Installing SunNet X.25 on a Diskless Workstation,” explains how to install SunNet X.25 on a diskless workstation.

Appendix F, “Cabling,” tells you how to build a synchronous null modem for use with SunNet X.25 and presents a pin-out diagram of an X.21-to-RS-449 converter.

Appendix G, “X.25 Routing,” describes how SunNet X.25 performs routing based on full or partial AEFs or X.121 addresses.

Appendix H, “X.121 Country Codes,” lists the three-digit country codes published by the CCITT. These codes are used in the “exchange” portion of an X.25 address.

Appendix I, “Acronyms,” is a list of acronyms with their definitions.

Appendix J, “Glossary,” lists X.25-related terms with their definitions.

Appendix K, “`gated` Man Pages,” is the `gated` man pages, which we provide to you as a convenience. The `gated` routing package, commonly used in the Internet world, is not supported by Sun.

Conventions Used in this Manual

This manual uses the following conventions:

- The acronym PSDN (Packet-Switched Data Network) is used to refer to any public or private packet-switched network that makes available to users interfaces that comply with the X.25 standard.
- We use the acronym HDLC (High-level Data Link Control) to refer the Link Access Protocol, Balanced (LAPB), which is a subset of HDLC that is used under the X.25 Packet Layer Protocol.
- Except where we refer to a specific model, we use the term “Sun workstation” to refer to the Sun-3™, Sun-4™, and SPARC™ (-station or -server) machines.
- Unless we distinguish between the two sets of numbers in text, we specify hexadecimal numbers with a prefix of 0x and decimal numbers without a prefix. For example, hexadecimal 10 is 0x10, while decimal 10 is 10.

We use the following typographic conventions:

`Typewriter font`
Represents what the system prints on your workstation screen and is used for program and file names.

`Boldface typewriter font`
Indicates literal user input, typically commands and responses to prompts that you can type in exactly as printed in this manual.

Boldface font
Emphasizes information within the text.

Italic font
Indicates variables or parameters that you replace with an appropriate word or string. Also used for emphasis.

`hostname%`
Represents your system's prompt for a non-privileged user's account.

`hostname#`
Represents your system's prompt for the `root` (super-user) account.

Gray-shaded
Contain text that represents interactive sessions. User input is indicated by **`boldface typewriter font`**.

Non-shaded
Contain text that represents part of a configuration file or program output.

Document Set

This manual is part of a document set that also includes the *SunNet X.25 Application Guide* and the *SunNet X.25 Programmer's Manual*.

1

Overview

This chapter presents an overview of the X.25 standard and introduces the SunNet X.25 product, including a description of its primary uses. This chapter also offers a summary of SunNet X.25 features.

1.1. How to Use This Manual

Table 1-1 lists the chapters and documents you require for installation and configuration, for each use of SunNet X.25.

Table 1-1 *Chapters Required for Each Use of SunNet X.25*

Product Application	*Chapters or Document Required*
Platform for SunNet OSI	3, 5 and *SunNet OSI Installation Guide*
Connect TCP/IP networks	3, 4, 5, and 6
Connect to the DDN	2, 3, 5, 6, and Appendix C
Packet service for user-supplied applications	3, 5, 6 and *SunNet X.25 Programmer's Manual*
Platform for PAD/X.29	3, 5, 6 and *SunNet X.25 Application Guide*
HDLC service	3, 5, 6, and 8

Chapter 9 has troubleshooting information for all uses of SunNet X.25.

1.2. Overview of the X.25 Recommendation

The following overview provides a general background on the X.25 recommendation. If you are already familiar with X.25, you can skip to section 1.3.

The X.25 recommendation defines three layers of protocol:

1. the packet layer
2. the frame layer
3. the physical layer

The physical layer defines the hardware interface and the lowest level signalling conventions. Refer to Figure 1-1 for a diagram of a packet data network showing the three DTE/DCE (Data Terminal Equipment/Data Circuit-terminating Equipment) interface levels. The following three subsections describe the three protocol layers in more detail.

The Physical Layer

The X.25 specification defines two alternatives at the physical level: X.21 and X.21 *bis*. X.21 supports more flexible control information exchange than X.21 *bis*. X.21 *bis* defines how RS-232-C or V.24 can substitute for X.21. X.21 *bis* is the most commonly used physical interface technique used today and the technique supported by SunNet X.25.

The Frame Level

The second layer of X.25 is the frame level. The X.25 specification for this layer allows either of the Link Access Procedures LAP or LAPB to be used for data interchange between the DTE and DCE. SunNet X.25 uses LAPB, as do most X.25 implementations. LAPB is a subset of High-level Data Link Control (HDLC). The X.25 use of HDLC is always full-duplex.

SunNet X.25 makes HDLC available independent of full X.25 for use as a point-to-point protocol. Its use is limited to full-duplex only.

The Packet Level

Layer 3 of X.25 is the packet level. This layer supports the concept of virtual circuits between DTEs. The packet level also supports addressing of DTEs and the multiplexing of fully reliable end-to-end circuits. PSDNs usually support both switched virtual circuits (SVCs) and permanent virtual circuits (PVCs). Switched circuits are dynamically created and destroyed. An SVC has a beginning that occurs when one DTE calls another, a middle that is the duplex data transfer stage, and an end that occurs when one DTE breaks the circuit. Permanent virtual circuits are created automatically when a DTE activates its link to the PSDN and are always in the data transfer stage.

Figure 1-1 *X.25 Protocol Levels*

1330

1.3. Introduction to SunNet X.25

SunNet X.25 allows Suns to reliably exchange data packets with other X.25-conformant systems over Packet-Switching Data Networks (PSDNs) or 802.*x* LANs using the X.25 international standard.

Along with support for the three levels of X.25, SunNet X.25 provides a Logical Link Control Type 2 (LLC2) interface so that you can run the X.25 Packet Layer over an 802.*x* or FDDI LAN.

In addition to implementing the protocols and services of the X.25 standard, SunNet X.25 also supports the CCITT X.29 and X.3 standards, enabling a workstation running SunNet X.25 to act as the server or user end of a virtual terminal connection.

After you install SunNet X.25 on a Sun workstation, that workstation becomes an X.25 gateway, available to all of the machines on a TCP/IP network. SunNet X.25 can support the PAD and X.29 server applications and OSI and TCP/IP clients at the same time as it maintains X.25 virtual circuits over a PSDN or LAN. Using these virtual circuits, the SunNet X.25 gateway can communicate with any DTE, Sun or non-Sun, accessible from that PSDN or on that LAN.

Figure 1-2 *General Representation of a Packet-Switched Data Network*

1327

1.4. Primary Uses for SunNet X.25

SunNet X.25 has the following primary uses:

- platform for SunNet OSI
- connecting TCP/IP networks
- connecting to the Defense Data Network (DDN)
- support for user-written applications
- platform for PAD/X.29 applications (supplied with SunNet X.25)

- running applications over HDLC

Figure 1-3 breaks down the primary uses, as they are supported over HDLC and LLC2.

Figure 1-3 *Primary Uses over HDLC and LLC2*

The following subsections cover each of the uses of SunNet X.25.

Platform for SunNet OSI

You can run SunNet X.25 under the Connection-Oriented Network Service (CONS) and the Connection-Less Network Service (CLNS) provided by SunNet OSI. In this use, SunNet X.25 provides a reliable packet service to the OSI Network Layer.

SunNet X.25 support for SunNet OSI is illustrated in Figure 1-4.

Figure 1-4 *SunNet X.25 as a Platform for SunNet OSI*

xxxx

Connecting TCP/IP Networks

SunNet X.25 allows you to use X.25 virtual circuits as point-to-point links for connecting remote TCP/IP networks over a PSDN. In this use, SunNet X.25 provides a transparent datagram service for carrying Internet Protocol (IP) packets over X.25/HDLC links. This capability allows SunNet X.25 to support SunOS routing software so that you can connect remote TCP/IP networks over a PSDN. (It is this same capability that allows SunNet X.25 to run carry OSI Network Layer packets and thereby support the SunNet OSI product.)

In connecting IP networks, SunNet X.25 performs a similar service to that provided by the SunNet Internetwork Router. In support of this use of the product, SunNet X.25 has an `x25mgr` program that manages virtual circuits for connecting remote TCP/IP networks.

SunNet X.25 provides no IP routing functions. Rather, it supports the IP routing that is part of the SunOS software and other IP implementations.

Figure 1-5 *SunNet X.25 to Connect TCP/IP Networks*

1328

Connecting to the DDN

SunNet X.25 provides the DDN X.25 Standard Service in support of Class A or B TCP/IP networks connecting to the DDN. This support is described in detail in Chapter 2.

Support for User-written Applications

The *SunNet X.25 Programmer's Manual* tells you how to write applications that use the X.25 packet level and HDLC (LAPB) frame level interfaces to SunNet OSI.

You can use SunNet X.25 as a packet service, to carry application protocols across a PSDN or LAN, either between machines running SunNet X.25 or between a SunNet X.25 machine and non-Sun machines. The product provides a programmatic interface based on the SunOS socket mechanism to this service. This interface is described in detail in the *SunNet X.25 Programmer's Manual*.

Examples of application protocols are file transfer, virtual terminal, and electronic mail.

An application running over SunNet X.25 can commmunicate, across the PSDN or on the LAN, with any other application using a compatible application protocol. The remote application need not be built over SunNet X.25 or the SunOS system.

In developing a SunNet X.25 application, you implement the application protocol as a SunOS user process, following the guidelines in the *SunNet X.25 Programmer's Manual*.

Figure 1-6 *Support for User-written Applications*

1329

Support for PAD and X.29 Server Applications

SunNet X.25 provides both X.3 user and X.29 server applications in support of virtual terminal sessions over a PSDN or LAN. The PAD application allows a Sun workstation user to login to a remote host as a terminal, over a PSDN or LAN. This application provides both the terminal emulation and PAD (packet assembler/disassembler) functions.

The X.29 server application allows a Sun workstation to act as a packet-mode host, able to maintain sessions with remote, character-mode terminals (virtual or real), over a PSDN or LAN.

These applications are documented in the *SunNet X.25 Application Guide*.

HDLC Service

SunNet X.25 supplies device driver interfaces, installed by `install.x25`, that support the running of SunOS applications over the HDLC frame level. SunNet X.25's LAPB/HDLC service provides reliable, flow-controlled transmission of HDLC frames.

SunNet X.25 provides a programmatic interface to the same HDLC interface used by SunOS applications. This interface is described in the *SunNet X.25 Programmer's Manual*.

Through a separate lower-level programmatic interface, you can also access the HDLC service at the frame level. This level provides HDLC framing without the LAPB protocol. It provides error detection through CRC checks, but not error correction, sequence checking, or flow control. The interface to this level is also documented in the *SunNet X.25 Programmer's Manual*.

1.5. Shared Links and Multiple Links

SunNet X.25 allows a single link to be shared among multiple uses of the product. For example, you can have PAD/X.29 traffic and IP/X.25 traffic running over the same link. The product also supports multiple serial (HDLC) links and multiple LLC2 links, with routing over these links.

1.6. Feature Summary

This section lists the overall features of the product. See Section 1.7 for a list of the 1984 X.25 features supported.

- Conforms to the 1980 and 1984 CCITT Recommendations for X.25
- Supports multiple physical links per gateway. You can have as many connections as serial ports and other resources allow. Supports routing on these links, based on AEFs or X.121 addresses. (See Appendix G.)
- Places no limits on the number of virtual circuits. You are limited only by machine resources and the configuration of your PSDN.
- Maximum data rates of:
  - 2.048 Mbits/sec. on HSI ports
  - 64 Kbits/sec. on MCP and SPARCstation CPU ports
  - 9.6 Kbits/sec. on Sun-3/50 CPU ports; 19.2 Kbits/sec. on CPU ports for other Sun-3 models
- X.29/X.3 packet-mode host and virtual terminal/ PAD support.
- Certification with major international PSDNs
- Standard network interface for kernel-resident protocol layers, such as CLNP, CONS, and IP
- Programmatic interface at socket level for X.25 programs and at device driver level for HDLC programs
- Supports the Address Extension Facility (AEF) for running the X.25 Packet Layer over LLC2 or HDLC
- Supports extended packet sequence numbering, non-standard default window sizes, and non-standard packet sizes
- Supports reverse charging, RPOA selection, flow control parameter negotiation, CUG selection, throughput class negotiation, and fast select on a per-call basis

- Supports large flow control windows (up to 127 packets) and large packet sizes (up to 1024 bytes) over both 1980- and 1984-compliant X.25 networks

1.7. 1984 Features

SunNet X.25 supports the following features of the 1984 X.25 recommendation:

- The following facilities:
  - Extended/Bilateral CUG (with outgoing access)
  - user data on Accept and Clear with fast select
  - network user identification
  - charging information facilities
  - Called Line address modified notification
  - call redirection notification
  - transit delay selection and indication
  - extended RPOA facility
  - calling AEF
  - called AEF
  - minimum throughput class
  - end-to-end transit delay
  - expedited data negotiation
  - non-X.25 facilities (in Call Request and Call Accepted packets)
- Up to 32 bytes of interrupt data
- Up to 109 bytes of facilities in Call Request and Call Accepted packets
- DTE-originated cause codes

SunNet X.25 does not support on-line facility registration and does not support the Multilink Protocol at the HDLC level.

2

Overview of DDN Support

SunNet X.25 supports the DDN X.25 Standard Service, via the `std` point-to-multipoint interface. This chapter gives an overview of the Sun implementation of X.25 Standard Service and provides information on connecting to the DDN.

2.1. DDN Architecture

The networks that make up the DDN have the same essential architecture: multiple networks are interconnected by a series of Packet-Switching Nodes (PSNs), which are connected to a backbone of high-speed communication lines. (The designation "Packet-Switching Node" has gradually superseded the older "Interface Message Processor" [IMP].)

The PSNs are controlled remotely by the Network Operating Center (NOC), which can bring the PSNs up, take them down, or reload them with new software over the network. Each PSN has one or more high-speed connections to other PSNs, and one or more local host connections, known as ports. Each port can be individually configured to communicate with the connected host using one of a number of protocols, such as 1822/HDH or X.25 Standard or Basic Service. (X.25 Standard Service has become the recommended protocol.) Often PSNs are located offsite, and a dedicated modem is used for communication between a local host and a remote PSN. Once a host has been physically connected to a PSN, and the Network Operating Center has brought the PSN port online, the host can send and receive TCP/IP datagrams to and from other hosts throughout the DDN.

2.2. How to Apply for Connection to the DDN

To join the DDN, you start by filling out a form, "Internet Number Template," which lets you request a DDN (Internet) address and an autonomous system number, and apply for connection to the DDN. This form is available from:

DDN Network Information Center
SRI International
333 Ravenswood Ave., Rm EJ291
Menlo Park, CA 94025

You can reach the DDN Network Information Center (DDN NIC) by phone at 800-235-3155.

2.3. Signaling Rates and Physical Interfaces Supported by the DDN

Sun has serial port options, described in Section 3.6, that support all of the rates and interfaces specified below. (Most options support rates that exceed the rates supported by the DDN.)

DDN PSNs offer three recommended† types of synchronous serial interfaces. With the appropriate Sun serial port option, Sun's X.25 software supports the maximum (and all lower) signaling rates supported for each type of interface:

RS-232-C
The CPU ports support the maximum signaling rate specified when using a modem eliminator (9.6 Kb/sec.) or when using a modem (14.4 Kb/sec.). These rates are also supported by the RS-232-C ports on the MCP board.

MIL-188-114 balanced (equiv. to RS-449/RS-422)
The MCP RS-449 ports support a rate of 64 Kb/sec., which is the highest rate supported without a local cable connection to the PSN. The HSI RS-449 ports support a rate of 100 Kb/sec., which the DDN makes available *with* a local cable connection.

V.35
The HSI V.35 ports support the maximum available rate of 100 Kb/sec.

Note that the two ports on an HSI board can be configured so that either or both are RS-449 or V.35 ports.

2.4. Protocol Options

The DDN supports the protocol stack illustrated in Figure 2-1. (This figure is derived from Figure 3-1 in Volume One of the *DDN Protocol Handbook*, 1985 edition.) Of the protocols in the figure, SunNet X.25 provides DDN X.25 Standard Service (in the shaded area).

† A fourth type of interface, MIL-188-114 (unbalanced), is available, but not recommended. This interface is equivalent to RS-449/RS-423, a lower speed alternative to RS-449/RS-422. Sun does not currently support RS-449/RS-423 in any of its serial port options.

Figure 2-1 *DDN Protocol Architecture and Sun Support*

1651

2.5. X.25 Standard Service

X.25 Standard Service is the DDN’s intended replacement for the 1822/HDH protocol option. Hosts running X.25 Standard Service can interoperate with all other hosts using X.25 Standard Service, plus hosts using the 1822/HDH Service, provided that all hosts use compatible protocols above the network-access protocols. The DoD protocol stack, including the network-access protocols, are illustrated in Figure 2-1.

Unlike X.25 Basic Service and X.25 as it is used in Public Data Networks, X.25 Standard Service is used only as a DTE-DCE (host-PSN) protocol, not an end-to-end (host-to-host) protocol. For end-to-end addressing beyond the DDN, X.25 Standard Service relies on the DoD-standard Internet Protocol (IP). Sun's implementation (as do other X.25 Standard Service implementations) converts the destination IP address in an outgoing packet, obtained from the SunOS IP routing table, to an X.121 address for transmission to the PSN and across the DDN. The address-conversion algorithm is described in section C.4. After traversing the DDN, the packet's IP address allows it to traverse multiple DoD-protocol gateways as it proceeds toward its ultimate destination.

With X.25 Standard Service, a virtual circuit is established only between the local host (e.g., the Sun X.25 machine) and the PSN. After the host and PSN have established the virtual circuit, the host can establish a TCP/IP connection to *any* host accessible from the PSN.

The DDN Program Management Office has indicated that X.25 Standard Service is the protocol of choice for access to the DDN. In the future, its use will be a requirement for a DDN connection.

The X.25 Standard Service implementation is compliant with the CCITT X.25 standard and FIPS 100/Federal Standard 1041.

2.6. DDN Routing

This section provides a broad overview of issues related to IP routing. For a useful source on IP routing, we recommend *Internetworking with TCP/IP: Principles, Protocols, and Architecture*, by Douglas Comer. The SunOS manual *System and Network Administration* has information on routing that is applicable to your needs within your own autonomous system.

Routing Options

One of the basic assumptions in a packet-switched network such as the DDN is that the network is not "fully connected," that is, all PSNs and directly connected hosts do not have direct links to all other PSNs and hosts in the network. In this context, the issues in administering the Sun X.25 software are choosing the best method for exchanging routing information within your autonomous system and choosing the best method for exchanging routing information with other autonomous systems.

When you use the X.25 Standard Service, you can use the `gated` software (see Appendix K), which allows you to use the Exterior Gateway Protocol (EGP) to exchange reachability information with other autonomous systems (including the set of core gateways) and the Routing Information Protocol (RIP) and the HELLO protocol to exchange routing information within your own autonomous system. A routing protocol, such as RIP, that is used within an autonomous system is referred to as an interior gateway protocol (IGP).

Figure 2-2 illustrates the complementary relationship of the EGP and IGP(s).

Figure 2-2 *Routing Information Exchange*

1652

Nodes *A*, *B*, and *C* are able to reach each other because of routing information obtained by means of either the Exterior Gateway Protocol (EGP) or an IGP. The two core gateways in Figure 2-2 exchange routes with each other by means of an interior gateway protocol and obtain routes from the non-core gateways via the EGP. The two non-core gateways exchange routes to their respective autonomous systems via the EGP. Within these autonomous systems, the non-core gateways exchange routing information with other gateways (not depicted) via an interior gateway protocol.

If your Sun DDN-gateway machine is a gateway to a very simple network, you might consider static routing, in which you preconfigure routes to all recognized gateways. Connections can be made only if *both* the Sun machine and the remote gateway have routing information about each other.

As an alternative to dynamic (EGP/IGP) and static routing, you can require that all incoming and outgoing connections must login to the Sun DDN-gateway machine. You control access to and from the DDN simply by restricting who can login to the gateway machine. The SunOS manual *System and Network*

Administration explains how to use the `/etc/passwd`, `/etc/group`, `/etc/hosts.equiv`, and `/.rhosts` files to restrict remote login access to a Sun machine.

DDN Core Gateways and Autonomous Systems

Routing of datagrams on the DDN is different from that used among SunOS systems, where routing information is circulated among all the hosts, using the `routed` daemon. On the DDN, only certain dedicated routing hosts keep track of all of the routes in the network of hosts connected directly to the PSNs (also known as the “core” or “backbone”). These routing hosts are known as *core gateways*.

For most host systems on the DDN, the Network Operating Center assigns one or more core gateways to each DDN host. Only the core gateways exchange routing information about the network and know how to get to any host directly connected to the DDN. Therefore, to connect to a DDN host on another IP network (for example MILNET to ARPANET), the Sun system must send the packet to a core gateway system. The core gateway chooses a gateway from the local IP network to the destination IP address, and then sends an ICMP redirect message to the Sun system. When the Sun host receives the redirect message, it establishes a route for that gateway in its routing tables, which it will use for all subsequent connections to that particular remote IP network.

Many DDN hosts are, in fact, gateways to IP networks that are not part of the DDN core. The core gateways have no information on which IP addresses may be reachable beyond these gateways, because in many cases they use a different routing scheme (like the SunOS `routed`) to keep track of routing information. These IP networks are commonly referred to as *autonomous systems*, because their internal operation is autonomous from the DDN core.

To enable autonomous systems to exchange IP packets among each other, the gateway machines must know how to route packets to other autonomous systems. You can accomplish this on the Sun DDN gateway by manually adding routing entries to the Sun machine’s routing table and to the routing table on the remote gateway machines. This approach is reasonable for one or two pairs of autonomous systems, but for full connectivity to all other autonomous systems, you should use a daemon running the Exterior Gateway Protocol (EGP) or the `gated` protocol. These protocols circulate reachability information between autonomous systems, allowing the gateway machine to route IP packets from any local IP network (that is part of the local autonomous system) to a remote IP network through the DDN using the DDN hosts as gateways. Configuration and use of the `gated` protocol is discussed in Appendix K.

2.7. Establishing Domains

In the context of the Internet’s domain name system and the Domain Name Service protocol (supported by SunOS software), a *domain* is a naming concept that allows a collection of machines to be administered as a single entity. Within a domain, all named objects, such as machines or users, must have different names. Two host machines in different domains can have the same name, just as two files in different directories can have the same name. A full pathname distinguishes two like-named files. Part or all of the domain name added to the hostname distinguishes two like-named hosts.

If you are connecting to the DDN, you will probably want to establish one or more domains within your network. The SunOS manual *System and Network Administration* explains the concept of domains and tells you how to apply to the DDN NIC for a domain name. In addition, the DDN NIC has a number of documents available that explain the concepts and implementation of domain naming. The address and phone number of the DDN NIC are given in Section 2.2.

3

Preparation for Using X.25

This chapter tells you the information you need before you install and configure SunNet X.25. To respond to the prompts in the SunNet X.25 installation script, `install.x25`, you must obtain certain items of information, either from your PSDN administration or from a system administrator of your LAN. This chapter provides background on these items. It also discusses serial port and modem requirements.

At the end of this chapter is a questionnaire that asks you to obtain all of the information you will need to complete configuration of SunNet X.25. Use this chapter, your PSDN contact person or LAN administrator, and Appendix A as references when you fill out the questionnaire. Appendix A presents parameter setting required by major PSDNs throughout the world.

NOTE: *If you intend to use* ***only*** *the HDLC service portion of SunNet X.25, you need only sections 3.3 through 3.7 in this chapter. After reading that, you can proceed to software installation, as described in Chapter 5.*

`install.x25` uses your responses to create a link configuration file for each link you have. The script also creates a configuration file for `x25mgr`, a program that manages virtual circuits when you use IP or OSI (CLNP) over X.25. After you finish with `install.x25`, you use the script `edit.x25params`, stored in `/usr/sunlink/x25`, to make changes to your link configuration file. You can use your favorite editor to make changes to your `x25mgr` configuration file, should that file require changes following installation.

Appendices B and C (for a DDN connection) contain a parameter-by-parameter description of the link configuration file. Appendix D contains descriptions of `x25mgr` configuration files used for the most common uses of SunNet X.25.

NOTE: *SunNet X.25 is shipped with an* `x25config` *program that reads your entries in an* `x25params` *file and configures your X.25 links. SunNet X.25 presents a programmatic interfaces that allow you to develop your own applications that do* ***not*** *use the* `x25config` *program. If you plan to use SunNet X.25 only for such an application, you need only sections 3.3 through 3.7 in this chapter. See the SunNet X.25 Programmer's Manual for a description of the X.25 programmatic interfaces. The* `x25config` *command is described in Section B.5.*

3.1. How Do You Plan to Use X.25?

SunNet X.25 supports the uses of X.25 discussed below. These uses are illustrated in Figure 3-1.

platform for SunNet OSI
If you plan to run SunNet X.25 under SunNet OSI, follow the software installation instructions in Chapter 5 then proceed to the *SunNet OSI Installation Guide*.

static or dynamic IP/X.25
You can use SunNet X.25 to connect geographically-separated TCP/IP LANs across a PSDN. In this use, X.25 software provides a virtual link between two distinct networks. SunNet X.25 provides no IP routing functions†. Rather, it supports the IP routing that is part of the SunOS software and other IP implementations.

You can create static or dynamic links between IP networks. In this context, “static” means that SunNet X.25 keeps the link up indefinitely. It can be brought down only by the remote host or by the network. The term “dynamic” means that SunNet X.25 brings a link up when there is a packet to send and brings the link down after a period of inactivity. Note that this use of dynamic and static is unrelated to switched and permanent, as those words are used to describe virtual circuits. IP/static links support both permanent and switched virtuals circuits; IP/dynamic links support only switched virtual circuits.

Considerations for using a static or a dynamic link are discussed below.

dynamic IP/DDN X.25 Standard Service
SunNet X.25 supports connections of Class A or B IP networks to the DDN using the DDN X.25 Standard Service. SunNet X.25 uses dynamic links to connect to the DDN. As stated above in the description of IP/X.25, this means that SunNet X.25 brings up a link on demand and takes it down when it is inactive.

SunNet X.25 does not support the 1822/HDH protocol for connecting to the DDN. The DDN X.25 Standard Service is identical to IP/X.25, as described above.

PAD/X.29 server
The `x29` program allows a machine running SunNet X.25 to act as a packet-switched host to remote asynchronous terminals that communicate with the host via a Packet Assembler/Disassembler (PAD). The `pad` program combines PAD and asynchronous terminal emulation functions, allowing you to log in to a remote X.29 host. If you are planning to run the X.29 or PAD applications that are shipped with the product, to respond to `install.x25` you need only know whether you will allow reverse-charged calls for X.29. The remaining X.29 (server) and PAD (user) configuration and use instructions are presented in the *SunNet X.25*

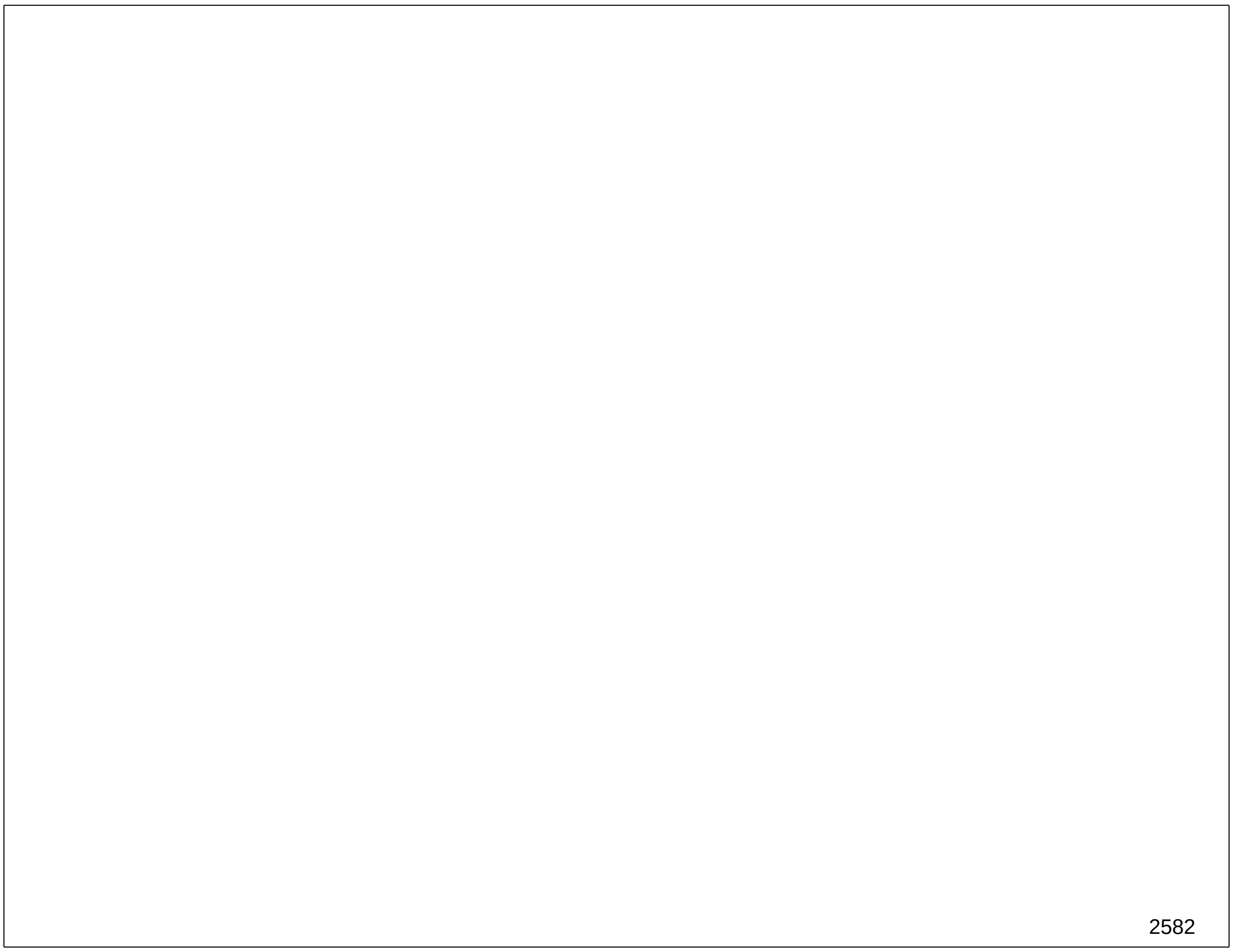
† SunNet X.25 is shipped with the `gated` routing information package, described in Appendix K. However, we provide `gated` only as a convenience for those who might want to use it. It is not a Sun-supported product.

Application Guide.

User-written applications

SunNet X.25 has programmatic interfaces that allow you to write X.25 applications that run over HDLC or LLC2. These applications can use the `x25config` program to configure a link and can use `x25mgr` to manage virtual circuits.

Figure 3-1 *Uses of SunNet X.25*



3.2. Static vs. Dynamic Links

If you use SunNet X.25 to connect TCP/IP networks, you have the choice of using static or dynamic links.

A static link is always up, unless the network or the remote side goes down. A static link uses a pair of point-to-point interfaces, analogous to the interfaces used by the SunNet Internetwork Router product. A dynamic link is brought up on demand (when SunNet X.25 has a packet to send) and brought down following a period of inactivity. A dynamic link uses a point-to-multipoint interface, similar to an Ethernet interface, with the significant difference that the point-to-multipoint interface does not have a broadcast capability.

Depending on the trade-off between connection setup costs and per-packet costs, static links are usually more expensive than dynamic when connecting across a subscriber PSDN. Exceptions to this occur when packet traffic is not continuous, but frequent, so that connection setup costs are high.

If you need to run `routed (8C)`, you must run the static service. If you can run `gated` (see Appendix K) or are using static routing (with the `route` command), you can run the static or the dynamic services. See Chapter 4 for more information on IP routing.

3.3. Installing System Files

In the course of installation, `install.x25` copies system files required by SunNet X.25 to your root hierarchy. If the script finds any files that require replacement, it saves the existing files before putting the new ones in place. The script renames your existing files by appending a period and a random process id number to the current name. For example, if `install.x25` uses the process id of 354, the file `/sys/netdlc/llc_conf.c` becomes `/sys/netdlc/llc_conf.c.354`.

3.4. Configuring Your Kernel

The `install.x25` script walks you through kernel configuration, asking you for the information it needs to build a new kernel containing the SunNet X.25 software. For the kernel configuration part of the script, you need to supply or be aware of the:

configuration file name

The name of the kernel configuration file to use as a base for building the new kernel. If you have the SunLink 6.0 X.25, the configuration file used for that products is a good choice.

A word of caution about `GENERIC` kernels on Sun-3s: the addition of the SunNet X.25 software options adds roughly 10% to the size of your existing, non-X.25 kernel. With certain combinations of Sun-3 machines and SunOS versions, kernels larger than 1.2 Mb do not boot reliably. To avoid such a problem, when building a new kernel, use one of the smaller configuration files supplied with SunOS, rather than `GENERIC`.

space in your `root` *partition*

As stated in the preceding paragraph, a kernel containing SunNet X.25 software adds to the size of a non-X.25 kernel. Compute your space requirements based on what you will need for a kernel that is 10% larger than your present `vmunix`. Use the `size` command to determine the size of your `/vmunix` file.

3.5. Disk Space Requirements

SunNet X.25 has the disk space requirements described in the Table 3-1. Figures are approximate. Note that these requirements are for free space. For example, the figures specified for space for a new kernel are for space in addition to the space occupied by the existing kernel.

Table 3-1 *Disk Space Requirements*

Hierarchy Name	*Space Required* *Sun-3*	*Sun-4 (including SPARC)*
`/usr`, or other destination partition	800 K	1000 K
``/usr/sys/`arch -k` ``	240 K	260 K
`/etc/sunlink/x25`	62 K	65 K
`/`, for new kernel	100 K	120 K

3.6. Serial Port Options and Data Rates

SunNet X.25 supports the synchronous serial port options listed below. Your choice depends on the type of interface to the PSDN (or DDN PSN) and the signaling rate to which you subscribe. When deciding between the CPU serial ports and the remaining options, cost might be a further consideration. Note that SunNet X.25 supports multiple physical links. The maximum signaling rates listed for each option might be in excess of the rates specified by your PSDN (or the DDN) for each type of interface.

CPU RS-232-C ports

Your workstation has one to four RS-232-C ports available for use with SunNet X.25. The maximum signaling rate for your port depends on your machine architecture. On a Sun-3/50, the CPU port does not reliably support a signaling rate greater than 9.6 Kb/sec. By contrast, a SPARCstation CPU port supports a rate of 64 Kb/sec. (The SPARCstation IPC does not have serial ports that support SunNet or SunLink products.) Obviously, maximum achievable data rates are affected by the load, both on other ports and on the CPU and other resources. Consult your hardware documentation for guidelines on maximum rates.

MCP ports

A single MCP board has four synchronous serial ports—two RS-232-C and two RS-449/RS-422. MCP ports support rates up to 64 Kb/sec.

HSI ports

A single HSI board has two synchronous serial ports, both of which you can configure to be either RS-449/RS-422 or V.35 interfaces. An HSI board supports a combined signaling rate of up to 4.096 Mbits/sec between the two ports. Each port supports signaling rates up to 2.048 Mbits/sec. For example, you can configure two ports to run at 2.048 Mbits/sec., or configure one port to run at 2.048 Mbits/sec. and the other at 64 Kbits/sec.

The MCP boards are VME-bus-compatible. Consult SunLink MCP and SunLink HSI documentation for more information on these products. See Section 2.3 for a discussion of signaling rates and interfaces supported by the DDN.

3.7. Modem Requirements

For a connection to a PSDN (or the DDN), you need a synchronous modem, a Channel Service Unit (CSU), or a direct connection. You also need a cable to the modem, CSU, or PSDN interface. For direct connections to the DDN, the Network Operations Center usually provides a cable from the PSN to your machine.

With remote PSDN connections, the modem (or equivalent interfacing equipment) must be compatible with that used at the remote end of the point-to-point circuit. The equipment must support synchronous signaling techniques and must be matched to the transmission facility. The equipment should support local and remote loopback to facilitate reliable installation and diagnostics. In addition, the cable between the communication port and the interfacing equipment must support the signals required for synchronous communication.

At some sites, the physical interface of your Sun serial port might differ from the interface provided by the network interfacing equipment. In such a situation, you must obtain a third-party protocol converter to achieve the necessary physical-interface compatibility.

3.8. Are You Connecting to a PSDN or to a LAN?

The current release of SunNet X.25 supports X.25 connections to LANs, via an LLC2 interface, in addition to direct connections to PSDNs. The sets of information you need for each type of configuration, one for LANs, one for PSDNs, are roughly the same, but there are significant differences, particularly in link-level parameters. In the following subsections, we discuss what is needed for each type of configuration.

Appendix B contains complete explanations of the parameters discussed in the following subsections.

Information Required If You Connect Directly to a PSDN

To respond to the prompts in `install.x25`, you must obtain the information listed below for each link you plan to support.

See Appendix A for parameter settings for some major public and private PSDNs in Europe, Asia, and North America.

baud rate
: What signaling rate are you using to connect to the PSDN? If you are using a synchronous modem, the modem determines the signaling rate. If you are using your Sun machine to supply clocking, your signaling rate is determined by the speed you specify in the `syncinit` command that initializes your serial port. (`install.x25` takes care of creating the required `syncinit` command.) Even if your modem or a remote machine supplies clocking, if you do not specify the correct signaling rate in a `syncinit` command, you will not be able to obtain accurate statistics from the `syncstat` program.

local X.121 address
: Find out your local X.121 address, including DNIC, NTN, and subaddress, if used.

1984 conformance
: SunNet X.25 supports both the 1980 and 1984 X.25 recommendations. You need to find out whether your PSDN supports the 1984 recommendation.

default throughput

This is the throughput class you subscribe to with your PSDN. These classes, listed below, correspond to the signaling rates at which you can access a PSDN.

Value	*Bit Rate*
3	75
4	150
5	300
6	600
7	1200
8	2400
9	4800
10	9600
11	19200
12	48000

Throughput class includes factors in addition to line speed, such as packet and window sizes and PSDN-specific parameters. For most applications, you should use the highest throughput class compatible with your network-access line speed.

LCN range

First, are you subscribing to any or all of one-way incoming, two-way, or one-way outgoing virtual circuits? For each of these, find out the range of logical channel numbers that the PSDN allocates for you.

Calling and Called Address Components

Some PSDNs exclude the DNIC portion of the calling address on local outgoing calls. Some PSDNs prepend a zero to the calling address for certain types of calls. You need to find out how your PSDN deals with calling and called addresses. Specifically:

- whether the PSDN expects the complete destination (called) address—i.e., DNIC, NTN, and subaddress—on outgoing calls
- whether the PSDN expects you to exclude the DNIC and/or the NTN in the source (calling) address on outgoing calls
- whether the PSDN expects you to prepend a 0 (or some other digit) to source or destination addresses, to indicate the presence of a DNIC in the calling or called addresses.

In addition to the items covered above you have to match the HDLC-level parameter settings with those of the PSDN. Table 3-2 lists the HDLC-level parameters with the default values supplied by `install.x25`.

The default settings supplied in the `x25params` file created by `install.x25` work for many installations. However, you should confirm these settings with your PSDN administration. If you need to make any changes, use the script `/usr/sunlink/x25/edit.x25params`. The user interface to this script is identical to the link configuration portion of `install.x25`.

Table 3-2 *Frame-level Parameters*

Parameter Name	*HDLC Default Value*	*LLC2 Default Value*	*Description*
link	normal	extended	for HDLC links, determines whether normal or extended mode is used.
poll_timeout	4500 msec	4500 msec	T1 - retransmission timer
ack_delay	2000 msec	500 msec	T2 - maximum delay before ACK
idle_timeout	18000 msec	10000 msec	T3 - maximum idle link time
busy_timeout	6000 msec	6000 msec	T4 - busy timer
reject_timeout	1000 msec	3000 msec	T5 - reject timer
ack_timeout	5000 msec	1000 msec	T6 - ACK timer
tick	500 ticks	500 ticks	resolution of timer
max_frame_size	134 bytes	1031 bytes	N1 - maximum frame size
max_retries	10 retries	10 retries	N2 - maximum retries of T1
window_size	7 frames	127 frames	K window size: number of frames primary can send to secondary before secondary acknowledges
x25 <*mode*>	`dte`	`dynamic`	link-level address (DTE/DCE mode)
lsap	NA	0x7E	For LLC2 links, link selector, used with your MAC address to make your LSAP address

Information Required If You are Using LLC2

If you run X.25 over LLC2, you need the following items that are discussed in the previous section:

- your X.121 address (this is often null when running over LLC2)
- default throughput (the value for this is not used if you are only running from DTE to DTE over a LAN; however, if you communicate with hosts on a PSDN via an X.25 Interworking Unit [IWU], you might require a specific default throughput value)
- range of logical channel numbers

The `install.x25` script does not prompt for the following items. However, they are important for LLC2 connections.

link selector (LSEL)

This even (i.e., not odd) hexadecimal number is the address of a Network Layer entity. Together with the MAC address for your system, it forms an LSAP address. Your system administrator should be able to give you this value. Your LLC2 link configuration file is shipped with a default value of 0x7E .†

† Using the programmatic interface described in the *SunNet X.25 Programmer's Manual*, you can write a

Poll-bit timer
A frame-level parameter that specifies the length of time, in milliseconds, that your host waits for a response to a command (poll) before initiating error recovery action. The P-bit timer is determined by the setting of the `poll_timeout` parameter in your LLC2 link configuration file. This parameter has a default value of 2000 msec for LLC2 links.

busy timer
A frame-level parameter that specifies the length of time, in milliseconds, that your host will wait for the clearance of a busy condition in the remote host. The busy timer is determined by the setting of the `busy_timeout` parameter in your LLC2 link configuration file. This parameter has a default value of 6000 msec.

reject timer
A frame-level parameter that specifies the length of time, in milliseconds, that your host will wait for a response to a Reject frame. The reject timer is determined by the setting of the `reject_timeout` parameter in your LLC2 link configuration file. This parameter has a default value of 6000 msec.

ACK timer
A frame-level parameter that specifies the length of time, in milliseconds, that your host will wait for an acknowledgement to one or more outstanding data frames. The ACK timer is determined by the setting of the `ack_timeout` parameter in your LLC2 link configuration file. This parameter has a default value of 6000 msec.

For these parameters and the remaining parameters in the standard LLC2 link configuration file, SunNet X.25 supplies default values that we think will work for most sites. See Table 3-2, above, for frame-level defaults for LLC2. If you do not know or are uncertain of the “correct” settings, we suggest that you start with the defaults in your link configuration file.

3.9. Shared Links and Multiple Links

SunNet X.25 allows multiple applications to use the same physical link, or multiple physical links, simultaneously. This feature is limited only in that you cannot use the `std0` interface—the interface used for dynamically opened and closed virtual circuits—for IP traffic over the DDN at the same time as you are using the interface for IP traffic to networks other than the DDN.

On a given machine, you can have one of either dynamic IP/X.25 or DDN/X.25 types of links. On the same machine, you can have one OSI(CLNP)/X.25 link. (DDN/X.25 and CLNP/X.25 links are always dynamic.) Over a single link you can have a dynamic IP/X.25 link, multiple static IP/X.25 links, and a CLNP/X.25 link. Over the same link, you can run OSI(CONS)/X.25 and PAD/X.29 traffic. The `x25mgr` program manages virtual circuits for IP/X.25, DDN/X.25, and CLNP/X.25. CONS and PAD perform their own virtual-circuit management.

---

program that listens on an LSEL value of your choosing.

If a single physical link does not provide adequate bandwidth to accommodate your applications, you can distribute multiple X.25 uses over multiple serial links. SunNet X.25 also allows you to have multiple LAN (LLC2) connections. For multiple links, SunNet X.25 has a routing function that routes packets to different links based on full or partial AEFs or X.121 addresses. This multiple-link routing is discussed in Appendix G.

NOTE: *SunNet X.25 does not support the 1984 X.25 Multilink feature.*

3.10. Information You Need for IP Routing

If you plan to use SunNet X.25 to connect two IP LANs, you need the following information for the `install.x25` script:

For dynamic IP/X.25

- The IP address of the dynamic IP interface, for which you can enter a hostname or a numeric address in Internet “dot” notation, e.g., 129.144.133.56 This address (or equivalent hostname) must have a network number that is different from the network number in the IP address of the primary network interface on your machine. On most machines, the primary network interface is the Ethernet or FDDI interface.
- The netmask used in your network (see the following subsection).
- An IP address (or hostname if it is listed in `/etc/hosts` or you use the NIS) and an X.121 address for each remote host to which you want to connect.

For DDN X.25 Standard Service

- The Class A or B IP address of the dynamic DDN interface, for which you can enter a hostname or a numeric address in Internet “dot” notation, e.g., 12.32.133.56. This address (or equivalent hostname) must have a network number that is different from the network number in the IP address of the primary network interface on your machine. On most machines, the primary network interface is the Ethernet or FDDI interface.
- The netmask used in your network (see the following subsection).

For Static IP/X.25 Links

- Your own IP address, for which you can enter your hostname or enter the address in Internet “dot” notation, e.g., 129.144.133.56.
- An IP address (or hostname if you use the NIS) and an X.121 address for each remote host to which you want to connect.
- Whether your host will be the caller or called machine.
- Whether you are using switched or permanent virtual circuits.

Recall from the section “Static vs. Dynamic Links,” above, that you cannot use `in.routed` over dynamic links.

When `install.x25` prompts for IP addresses, you can respond with a hostname rather than a numerical address. Such a hostname must be listed in a local `/etc/hosts` file or registered in the NIS's `hosts.byname` database. See the SunOS manual *System and Network Administration* for information on setting up and maintaining NIS databases.

3.11. Netmasks and Subnetting

The netmask is used to establish subnetting in your network. Netmask values are of the form `255.`*nnn.nnn.nnn* where *nnn* is a decimal number in the range from 0-255. When `install.x25` prompts for a netmask, you can respond with a `+`, rather than a string of digits. The `+` tells the script to look in a local `/etc/netmasks` file or in the network information service's `netmasks.byaddr` database for the netmask value. See the SunOS manual *System and Network Administration* for details on netmasks and subnetting.

Installation Questionnaire

Product Use

How will you use SunNet X.25?

Platform for SunNet OSI ________________
Connect TCP/IP networks ________________
Connect to the DDN ________________
Support for user-written applications ________________
PAD/X.29 server ________________
Running applications over HDLC ________________
Other use (please specify) ________________

Operating System Information

SunOS version number: ________________

Kernel configuration file name: ________________

Serial Port and Link Information

Serial Interface(s) (e.g., CPU RS-232-C, MCP RS-449): ________________

Line Interface Equipment (e.g., null modem, synchronous modem): ________________

Signaling rate: ________________

Number of links: ________________

Will you run multiple applications over a single link? ________________

Service Provider Information

Connecting to PSDN (via HDLC) or LAN (via LLC2)? ________________

Network name (if connecting to a PSDN): ________________

Does the network you specified in the previous question support the 1984 X.25 recommendation? ________________

HDLC-level Parameters

Maximum I-frame size (in bytes): ________________

Extended Control Parameter (normal or extended): ________________

Window size: ________________

(For null modem connections only) Is the local host a DTE or DCE? ________________

LLC2-level Parameters

Link selector (LSEL): ________________

P (poll)-bit timer: ________________

busy timer: ________________

reject timer: ________________

ACK timer: ________________

X.25 (Packet)-level Parameters

Local X.121 address (usually "null" for LLC2 links):

DNIC: ________________
NTN: ________________
Subaddress: ________________

Is all or part of the X.121 address used in a call request? ________________

Is a prefix used for long-distance calls? ________________

Range of packet sequence numbers (usually 8): ________________

Window size for all virtual circuits: ________________

Range of logical channel numbers:

For one-way incoming: ________________
For two-way: ________________
For one-way outgoing: ________________

Set D-bit (for end-to-end acknowledgment)? ________________

Default throughput (can be "unspecified"): ________________

4

Connecting TCP/IP Networks

SunNet X.25 can perform as a transparent datagram transmission service, allowing it to support the interconnection† of TCP/IP LANs across a PSDN or a null modem link. This chapter explains how to assign IP (Internet Protocol) addresses to X.25 point-to-point and point-to-multipoint interfaces and gives examples of `route` commands you use to establish routing across an X.25 link.

This chapter is not intended to be a definitive source on routing between IP networks. The section entitled "Setting Up a Router to Join Two Networks" in the SunOS manual *System and Network Administration* explains how to set up your IP routing gateway.

4.1. Terminology

In this chapter, we use "static" and "dynamic" in two different contexts:

- With regard to X.25 interfaces, static interfaces are always up, while dynamic interfaces go up on demand and down on inactivity.
- With regard to routing, static routing implies routing entries that you enter with the `route` command and that remain in place until reboot. Dynamic routing implies the use of a protocol, such as `routed`, that automatically propagates and accepts routing information. Routing entries installed by `routed` have a limited lifetime.

Another distinction to be clarified is that between a physical link and a virtual circuit. A single link can carry one or more virtual circuits. SunNet X.25 provides two different types of interfaces to a physical link, `xvc` and `std0`. The `xvc` interface supports a single virtual circuit at one time; the `std0` interface supports multiple virtual circuits at once. These interfaces are described in detail in the following section.

4.2. Point-to-Point and Point-to-Multipoint Interfaces

For the interconnection of IP networks, SunNet X.25 supports both a point-to-point interface, `xvc`<*n*>, and a point-to-multipoint interface, `std0`. You can have a single `std0` interface used for IP traffic on one machine. A single link can support multiple `xvc`<*n*> interfaces (numbered consecutively, starting with zero). A link can be shared among an `std0` interface and one or more `xvc` interfaces.

---

† The protocol used is described in the document *RFC 877, A Standard for the Transmission of IP Datagrams Over Public Data Networks*.

An `xvc` interface is considered a *static* interface because it is always up, unless the network or the remote host goes down. Also, for as long as the link is up, a given `xvc` interface is always associated with a single, remote point-to-point interface.

By contrast with an `xvc` interface, an `std0` interface is *dynamic*. That is, when running over `std0,` SunNet X.25 brings up the link on demand and brings it down after a period of inactivity. Also, the `std0` interface can be associated with multiple, different remote interfaces.

When the `x25mgr` program (see Appendix D) invokes the `ifconfig` program for an `xvc` interface, `ifconfig` associates that interface with the source and destination IP addresses for that link. For an `std0` interface, `ifconfig` associates `std0` with a source IP host address and the network number of that host address. For `std0`, IP software creates a routing entry, establishing the local network interface as a gateway with the network designated by the local host's network number as the destination.

4.3. Tasks to Set Up IP Routing

If you plan to use SunNet X.25 to interconnect TCP/IP networks, you must:

- decide on the use of one or more `xvc` interfaces or an `std0` interface
- assign an IP address to the X.25 interface
- decide on a routing method—static using the `route` command, dynamic using `in.routed`, or dynamic using the `gated` package

Your choice of whether to use a static or dynamic interface is largely a matter of cost. A static link has relatively higher (than dynamic link) connection-time costs, but almost no connection-setup costs. By contrast, a dynamic link has potentially lower connection-time costs, but higher connection-setup costs. If a PSDN has low connection-time charges and high connection-setup charges, a static link might be cheaper than a dynamic. Conversely, if a PSDN has high connection-time charges, but does not charge much to set up a call, a dynamic link might be the less costly alternative.

A further consideration in deciding between the two types of links is the routing method you use:

- You cannot run `routed` over a dynamic link.
- You can run configure `gated` to run over a dynamic link.
- You can run `routed` or `gated`, or use static routing with the `route` command, over a static link.

How you assign an IP address depends on whether you use an `xvc` or `std0` interface and, if you use an `xvc` interface, whether you use `in.routed`. Specifically:

- If you use an `xvc` interface:
  - If you use static routing or run the routing protocols in the `gated` package, you must assign an IP address to `xvc` that has a network number that is unique to the point-to-point link. This network number must

match the network number of the remote xvc interface. Under the conditions described here, IP routing software requires that the network number for the xvc interface be different from the network number of the primary interface.

- If you use the in.routed that is shipped with the SunOS software and your SunOS version is earlier than 4.1, you must assign an IP address to xvc that has a network number that is unique to the point-to-point link. This network number must match the network number of the remote xvc interface. Under the conditions described here, IP routing software requires that the network number for the xvc interface be different from the network number of the primary interface.

  If you are running SunOS 4.1 or later, you can assign to xvc the IP address of the primary (usually the Ethernet) interface on your machine. Alternatively, for the 4.1-or-later users, you can follow the pre-4.1 model and assign a different network number to xvc.

- If you use the std0 interface, you must assign an IP address to std0 that has a network number that is different from the network number of the LAN interface. If this network number matches the network number of the remote std0 interface, you do not have to add a routing entry to reach the remote side. Otherwise, you must use route to add a remote host or network entry.

4.4. Assigning IP Addresses

This section discusses the assignment of IP addresses for different combinations of X.25 interfaces and routing methods.

Assigning Machine-unique Network Numbers

If you use an std0 interface or have multiple xvc interfaces, you must use the addressing scheme described in this subsection. If you use the dynamic routing protocol implemented in in.routed and use one xvc interface, you must use the scheme described here if your SunOS version is earlier than 4.1. If your SunOS version is 4.1 or later, you have the option of using the scheme described here or the scheme described in the following subsection. If you use static routing, using routes created with the route command, or use the routing protocols in the gated package, you must use the scheme described here.

In the addressing scheme required by in.routed before (and still supported by) SunOS 4.1 each network interface of each host has a unique IP address. Network interfaces include a machine’s Ethernet or FDDI interface(s) as well as the std0 or xvc interfaces used by SunNet X.25. An IP address consists of a network number, to identify a specific network, and a host number, to identify an individual machine.

A routing gateway has at least two network interfaces. A routing gateway that uses SunNet X.25 has at least one X.25 interface (std0 and/or xvc<*n*>) for each link it uses for IP routing.

Note that the network numbers of the X.25 interfaces must be 1) different from the network numbers** of either of the networks being connected and 2) the same as the network number of the remote X.25 interface. If you assign IP addresses with machine-unique network numbers to your X.25 interface(s), you must perform the following steps:

- You must specify these additional IP addresses in the local `/etc/hosts` files or in the NIS's `hosts.byaddr` database.
- You must enter the additional network number in the `/etc/networks` file or in the network information service's `networks.byaddr` database, along with the network numbers of the networks you are linking. In effect, the X.25 link makes up its own tiny network, with the link itself having its own network number and the X.25 interfaces having unique host numbers on that network.

Example: Different Network Numbers

A network in Birmingham has the following host table:

```
192.9.203.1    bozo
192.9.203.2    bilbo
192.9.203.3    bonzo
```

A network in Framingham has the following host table:

```
192.9.201.1    frodo
192.9.201.2    freddy
192.9.201.7    fajita
```

The network number for Birmingham is `192.9.203`, while the network number for Framingham is `192.9.201`. These networks are illustrated in Figure 4-1.

**To obtain a unique network number contact:
DDN Network Information Center
SRI International
Room EJ217
333 Ravenswood Avenue
Menlo Park, CA 94025
(800) 235-3155

Figure 4-1 *IP Networks with X.25 Link that has Unique Network Number*

xxxx

The X.25 gateway machines, `bilbo` and `frodo` in our example, have X.25 interfaces with IP addresses in addition to the IP addresses assigned to their Ethernet interfaces. We merge the two host files, then add entries for the X.25 interfaces at both ends of the X.25 link. By convention, we recommend a name that uses the hostname followed by `–x25` and a letter, to designate which X.25 link you are using. (Since SunNet X.25 supports multiple links, a single gateway can support multiple connections to different remote Ethernets.) The merged host file contains:

```
192.9.203.1    bozo
192.9.203.2    bilbo
192.9.203.3    bonzo
192.9.201.1    frodo
192.9.201.2    freddy
192.9.201.7    fajita
192.9.202.1    bilbo-x25a
192.9.202.2    frodo-x25a
```

Note that the network number portion of the new IP addresses, `192.9.202`, differs from the network numbers of the two networks being connected.

The `/etc/networks` files for both `bilbo` and `frodo` contain:

```
Birmingham      192.9.200
Framingham      192.9.203
bilbo-frodo     192.9.202
```

In `install.x25` (or `install.osi`), when prompted for the address of the local X.25 interface, on `bilbo`, enter `bilbo-x25a,` and on `frodo`, enter `frodo-x25a`. The names `bilbo-x25a` and `frodo-x25a` are the symbolic equivalents of the IP addresses for the X.25 interfaces on `bilbo` and `frodo`.

Assigning Primary Interface Address to `xvc` Interface

If you have a single `xvc` interface, plan to use `in.routed`, and are running SunOS 4.1 or later, you can assign to the `xvc` interface the same IP address as the primary interface of the machine on which SunNet X.25 resides. (A machine's primary interface is usually its 802.*x* interface.) The address of the primary interface is the address associated with a machine's hostname, as recorded in the `/etc/hosts` file.

If you use the same IP address for the `xvc` and primary interfaces, you must add the `-private` flag to the `ifconfig` command in `/etc/sunlink/x25/x25manager.configuration`.

Example: Sharing Primary Interface Address

Consider the following example of two Ethernets that you intend to connect by an `xvc` link. All of the addresses in the example are addresses of the primary interface of each machine.

A machine named `bilbo` in a network in Birmingham has the following machines listed in its `/etc/hosts` file:

```
192.9.200.1    bozo
192.9.200.2    bilbo
192.9.200.3    bonzo
```

...while a machine named `frodo` in a network in Framingham has the following machines in its `/etc/hosts`file:

```
192.9.201.1    frodo
192.9.201.2    freddy
192.9.201.7    fajita
```

These networks are illustrated in Figure 4-2.

Figure 4-2 *IP Networks with X.25 Link that Shares Network Number*

xxxx

To link the two networks you must first ensure that the two networks use different network numbers. In the example networks, Birmingham has a network number of 192.9.200; Framingham has a network number of 192.9.201. Next, merge the two `hosts` files. The final `hosts` file for `bilbo` and `frodo` looks like this:

```
192.9.200.1    bozo
192.9.200.2    bilbo
192.9.200.3    bonzo
192.9.201.1    frodo
192.9.201.2    freddy
192.9.201.7    fajita
```

The `/etc/networks` files for both machines contain:

```
Birmingham      192.9.200
Framingham      192.9.201
```

If you use the NIS facility, use `ypmake` command, as documented in *System and Network Administration*, to make the merged `hosts` file available to the entire network.

In `install.x25` (or `install.osi`), when prompted for the address of the local X.25 interface, press Enter to accept the default, the hostname of the local machine, which is a symbolic representation of the numeric IP address. (On `bilbo`, this is `bilbo`, and on `frodo`, `frodo`.)

4.5. Routing Methods

You can use the methods listed below for propagating routing information within the TCP/IP networks interconnected with SunNet X.25. Note that these methods apply *within* TCP/IP networks, not across the X.25 link.

`in.routed`
: Shipped with SunOS. This daemon, stored in `/usr/etc`, is invoked at boot time from a line in `/etc/rc.local`. The daemon implements the Xerox NS Routing Information Protocol (RIP) commonly used in the UNIX† 4.3 BSD community. The `routed` software does not propagate routing information over an `std0` interface. See `routed (8C)` in the *Commands Reference Manual* that is part of SunOS documentation for a description of this daemon.

`gated`
: A package of routing protocols (RIP, HELLO, and Exterior Gateway Protocol [EGP]) that is made available with SunNet X.25, but not supported by Sun. See Appendix K for a description of this package. Generally, `gated` is more configurable than `routed`. You can use the EGP portion of `gated` over an `std0` interface.

static routing
: You can insert static routes using the `route (8C)` command, stored in `/usr/etc`. To avoid potentially conflicting routes, on a given machine, you should not use static routing in conjunction with one of the dynamic routing protocols. Static routing is suitable for both `xvc` and `std0` interfaces. See the documentation for the `route` command in `route(8C)` in

† UNIX is a registered trademark of UNIX System Laboratories, Inc.

the SunOS *Commands Reference Manual*.

You can run `routed` only over a static (`xvc`-to-`xvc`) link, not over a link that uses `std0`. Your choice of X.25 link (`xvc` or `std0`) does not affect the choice of routing protocol (or static routing) you use *within* the TCP/IP networks you are connecting.

4.6. Static vs. Dynamic Routing

Given the possible variations in TCP/IP internetworks, it is difficult for us to establish fixed guidelines that would apply to your decision on whether you should use static, dynamic, or a combination of static and dynamic routing in your internetwork. However, we can offer some facts about each type of routing:

- Dynamic routing is far easier to administer than static. In fact, beyond small internetworks, using only static routing is nearly impossible.
- The CPU overhead of running `in.routed` on a host is not great. However, the routing table maintained by the daemon takes up much more space in the kernel than do most static routing tables. (Use the `-r` option of `netstat (8C)` to view the current IP routing table.)
- The overhead of running `in.routed` on a router can be significant, particularly in a large internetwork.
- With the use of the `-q` (quiet) option of `in.routed` and active and passive gateway entries in `/etc/gateways`, you can limit the scope of `in.routed`'s broadcasts. (See the man page for `routed (8C)` for a description of the `gateways` file.) The protocol implementations in the `gated` package give you a finer degree of control over the propagation of routing information than does `in.routed`, at the cost of somewhat more complex configuration.
- Within a given TCP/IP network, you can use static routing in combination with dynamic routing. One way of doing this is to use `route` to create on hosts in a network a default static route to a router that is running `in.routed`. For example, assume a router named `gandalf` on your local network. On your host, in `/etc/rc.local`, you comment out the line to start `in.routed` and enter a line such as:

```
# command to create static route to default router
/usr/etc/route add default gandalf 1
```

  With such a method, hosts save on the kernel space and `in.routed` overhead. However, the router (`gandalf`, in our example) becomes a single point of failure for the network. This becomes very important in an environment in which NFS is heavily used. If the router goes down, hosts lose routes to machines from which they are mounting directories, thereby losing access to those directories.
- There can be an element of dynamism in static routing. A machine with static routes can acquire additional routes, from ICMP redirects. (These routes are indicated by a `D` in `netstat -r` output.) However, unlike

routes acquired by `in.routed`, routes obtained from redirects do not timeout.

4.7. Establishing a Route over an X.25 Link

This section discusses methods for establishing IP routes over an X.25 link.

Static Routes

If you use the `std0` interface, after the `ifconfig` command is invoked from your `/etc/sunlink/x25/x25manager.configuration` file, you obtain a routing entry such as the following:

```
hostname% netstat -r
Routing tables
Destination    Gateway     Flags  Refcnt Use    Interface
bilbo-frodo    bilbo-x25a  U      0      2804   std0
```

where `bilbo-x25a` is the symbolic name of the numeric IP address of `std0` and `bilbo-frodo` is the network name of your X.25 link. Such an entry gives you a route to a directly connected machine at the remote end of the link.

If you use static routing over an `xvc` link, you must enter `route` commands to establish a route to the remote end. The following example uses the example network in section 4.4 and uses IP addresses with unique network numbers for the `xvc` interfaces. As `root`, enter the following `route` commands on `bilbo` to establish a route to `frodo`.

```
bilbo# /usr/etc/route add bilbo-x25a bilbo-x25a 1
bilbo# /usr/etc/route add frodo-x25a bilbo-x25a 1
```

On `frodo`, enter the following commands to establish a route to `bilbo`:

```
frodo# /usr/etc/route add frodo-x25a frodo-x25a 1
frodo# /usr/etc/route add bilbo-x25a frodo-x25a 1
```

The number `1` at the end of each `route` command line specifies that the remote host is one hop away from the local machine.

If you assign to `xvc` the address of the host's primary interface, you do not need to enter a command to establish a route from an interface to itself. Using this method of address assignment, enter the commands below. On `bilbo`, to establish a route to `frodo`, enter:

```
hostname# /usr/etc/route add frodo bilbo 1
```

On `frodo`, to establish a route to `bilbo` enter:

```
hostname# /usr/etc/route add bilbo frodo 1
```

If you are using `in.routed` over an `xvc` link, you obtain routes in both directions over the X.25 link automatically, unless you run `in.routed` with the `-q` option. If you run the daemon with `-q`, you must enter `route` commands to add routes across the link, as described above.

Dynamic Routes

Unless you use `in.routed` without the `-q` option on machines at both ends of an `xvc` link, `in.routed` automatically propagates routing information over the X.25 link. Such information includes the routes in both directions over the X.25 link. Similarly, with `gated`, unless you configure the package to not transmit routing information, the daemon propagates routing information over the X.25 link.

If your router has only X.25 (and no LAN) interfaces or if you have an X.25 interface that has a machine-unique network number, you must add a loopback route to the local X.25 interface if you plan to propagate routing information over the X.25 link. To do this, you enter a command such as the following:

```
hostname# route add host frodo-x25a localhost 0
```

where `frodo-x25a` is your X.25 interface. Use of such a command presumes there is an NIS hosts.byname or `/etc/hosts` entry for `frodo-x25a`.

5

Software Installation

This chapter tells you how to install the SunNet X.25 software. You do this through an easy-to-use script that is shipped with the SunOS. All of the procedures described here assume that you are logged on as `root`. See Appendix E for instructions on installing SunNet X.25 on a diskless workstation.

5.1. Installation Notes

To accommodate SunNet X.25 software, you need approximately 800 kbytes of free disk space in a destination partition for each machine architecture supported on your machine. A machine's architecture is its type, such as Sun-4 or Sun-3. You obtain a machine's architecture with the `arch` command. Standalone (non-server) machines support only their own architectures. Server machines support their own architectures and, potentially, other architectures. For example a Sun-4 server can support the Sun-4 and Sun-3 architectures.

Before starting the script, make sure you have a symbolic link from `/export/exec/`*arch* to the `/usr` directory on your machine. (*arch* is a directory name, such as `sun4`, that corresponds to the return from the `arch` command.) This link should have been created for you when you installed your operating system. The link allows the script to place the SunNet X.25 files for *your* machine type in `/usr/sunlink`. The files for other machine types, if needed, are installed in the `/export/exec` directories. To see if you have the link, enter the following commands:

```
hostname# cd /export/exec
hostname# ls -l
```

You should see output similar to that shown below. Note that an `l` as the first character in an output line indicates that a file is a symbolic link.

```
lrwxrwxrwx  1 root       4 May 24 03:46 sun4 -> /usr
```

The link name, `sun4`, corresponds to your machine type.

If the link does not exist, enter the following commands to create it:

```
hostname# cd /export/exec
hostname# ln -s /usr sun4
```

The preceding command creates a link for a Sun-4, including SPARC machines. Use the link name appropriate for your machine, for example, `sun3`, for a Sun-3.

5.2. Directories Created

The `extract_unbundled` script creates the directories `sync` and `x25` under `/usr/sunlink`. The script allows you to install SunNet X.25 software in a partition different from `/usr`. If you select, for example, `/home` instead of `/usr`, the parent directory will be `/home/x25/‘arch‘/sunlink/x25` rather than `/usr/sunlink/x25`.

The product configuration script, `install.x25`, described in Chapter 6, creates a hierarchy under `/etc/sunlink/x25` and installs system files under `/sys`.

5.3. Running `extract_unbundled`

`extract_unbundled` is the first of a series of scripts that prompt you to enter certain parameters and, directed by your responses, copy the SunNet X.25 software from the distribution tape to an appropriate directory. It is the only script you must directly invoke; each script automatically passes control to the next in line. This section first describes the functions of each script and then shows the scripts’ prompts and discusses your possible responses.

NOTE: *If you are installing SunNet X.25 on a diskless workstation, you must run* `extract_unbundled` *on the NFS server for that diskless machine. See Appendix E for the procedure for installing SunNet X.25 on a diskless machine.*

Script Introduction

The following scripts extract and install the SunNet X.25 software from the distribution tape.

`/usr/etc/extract_unbundled`

This script, which is shipped with the SunOS, prompts you for the location of your tape drive and the device name. Once the script locates the tape, it displays the contents of the first file—the copyright statement—and asks if you want to continue the installation.

If you reply `y`, the script extracts the two `/usr/tmp/unbundled` scripts from the tape and passes control to the `/usr/tmp/unbundled/install_unbundled` script.

`/usr/tmp/unbundled/install_unbundled`

This script creates the log `/usr/tmp/unbundled/7.0_X25.log`. All output created by the installation process goes to this log. (You can use the log to re-create your installation later.) Once the log is created, the script passes control to `/usr/tmp/unbundled/install_SunLink`.

`/usr/tmp/unbundled/install_SunLink`

This script verifies that the SunOS running on your machine is compatible with the SunNet X.25 software. It then extracts the software appropriate for your machine and any machine types your machine supports as a server. When all files have been extracted, the script appends the copyright statement to `/usr/tmp/unbundled/Unbundled_Inventory`.

Prompts and Responses

This section explains the installation scripts' prompts, in the order in which you encounter them, and discusses your possible responses.

Enter the following command:

```
hostname# /usr/etc/extract_unbundled
```

You receive the following prompts:

```
Enter tape drive location ? (local | remote):
```

If you respond with *remote*, the script prompts you to enter the host name of the remote machine with a tape drive. Your machine must be listed in the `/.rhosts` file of the remote machine to obtain access to its tape drive. If the remote drive cannot be accessed or located, the script returns you to the operating system.

```
Enter Device Name (e.g. st0, mt0, ar0) : /dev/r
```

Enter the name of your tape device in the form suggested in the prompt. Following this, this script reminds you to mount your distribution tape. If the script cannot access your tape, it displays a message to that effect and returns you to the operating system. After the script accesses the tape, it displays a copyright notice and asks if you want to continue. Entering `n` returns you to the operating system.

In succession, the script then displays:

- an optional brief description (`Installs SunNet X.25`).
- a reminder of the software and hardware requirements and options for SunNet X.25
- the time installation will take
- the current free disk space and the space requirements for SunNet X.25 (800 kbytes per architecture)
- SunOS compatibility requirements

Following each of the above displays, the script asks if you want to continue. Entering `n` returns you to the operating system. If you continue, the script prompts:

```
Enter system type [standalone | server]:
```

`standalone` means that your machine is a private workstation that is not acting as a server. If you select standalone, the script reports the amount of free space and that the destination directory is `/usr`, then asks if you want to change to a different destination directory.

Enter `server` if your machine is acting as a server to clients on the network.

```
Enter server type [homo | heter]:
```

This prompt is displayed only if you entered `server` to the previous prompt.

`homo` indicates homogeneous: all the clients are the same machine type as the machine on which you are installing SunNet X.25. For example, if you are currently installing on a Sun-3 server, *all* of the clients must also be Sun-3s.

`heter` indicates heterogeneous: the clients are or may be of different machine types (for example, Sun-4 clients with a Sun-3 server) as the machine on which you are installing SunNet X.25.

For heterogeneous servers, the script prompts you for client machine types (`Will this be for a Sun# client?`), checks that a `/export/exec/`*arch* directory exists, and reports the amount of free space. If `/export/exec/`*arch* does not exist, the script returns you to the operating system.

For both servers and standalone machines, the script looks for an existing version of the SunNet X.25 software in the `/export/exec/sun4/sunlink/x25` and `/export/exec/sun4/sunlink/sync` directories. If it finds an existing version, it requests your permission to remove the `x25` and `sync` directories.

For standalone machines and homogeneous servers, the script checks for sufficient space in `/export/exec/‘arch‘` (which is a link to `/usr`). If there is sufficient space, it offers `/usr` as the destination partition and asks if you want to change destination partitions. If you want to accept the default, i.e., the `/usr` partition, enter `n`.

CAUTION: *To this point in the script you have been entering* `y` *to continue. Do not unthinkingly enter* `y` *to the prompt for changing your destination partition. If you enter* `y` *when you meant* `n`*, enter* `/usr` *when the script prompts you for a different destination partition.*

If there is not sufficient space in `/usr`, the script displays available partitions with their free space and asks if you want to change destination partitions. Enter the name of the new destination partition. The script asks if you want to create soft links from `/export/exec/sun4/sunlink/x25` and `/export/exec/sun4/sunlink/sync` to the new partition. Enter `y` to create the links.

After indicating once more that you want to continue, the script extracts the files from the distribution tape and copies them to the destination partition.

When you receive the message that the installation is complete, you are ready to proceed with product configuration.

5.4. Exporting Product Software to Clients

If the machine on which you install SunNet X.25 software is set up to export the `/usr` or `/export/exec/‘arch‘` directories, users of diskfull client machines can mount the hierarchy containing the product software. (The procedure for diskless clients is covered in Appendix E.)

To mount the appropriate hierarchy, as `root`, a user of a diskfull client machine creates a mount point and mounts the appropriate hierarchy as follows:

```
hostname# mkdir /usr/sunlink
hostname# mount -o soft,ro,bg,intr nfs_server:/usr/sunlink /usr/sunlink
```

nfs_server is the name of the machine on which you ran `extract_unbundled`.

Instead of the above `mount` command, you can add an entry such as the following in your `/etc/fstab` file:

```
nfs_server:/usr/sunlink /usr/sunlink nfs soft,ro,bg,intr 0 0
```

With such an entry, you can invoke the `mount -a` command to mount all of the entries in your `fstab` file.

With the `/usr/sunlink` hierarchy mounted, users of diskfull clients can perform the configuration steps described in the remaining chapters of this manual.

5.5. The Next Step

If you are using SunNet X.25 with SunNet OSI, see the *SunNet OSI Installation Guide*. Otherwise, proceed to Chapter 6 for configuration instructions.

6

X.25 Configuration

This chapter tells you how to configure SunNet X.25 for all uses of the product *except* its use as a platform for SunNet OSI. If you are using SunNet X.25 with SunNet OSI, after extracting the SunNet X.25 product tape, as instructed in Chapter 5, proceed to the *SunNet OSI Installation Guide*.

The various uses of SunNet X.25 are explained in Chapters 1 and 3.

You perform all of the configuration required for X.25 and all kernel configuration within the SunNet X.25 installation script, `install.x25`, stored in `/usr/sunlink/x25`.

NOTE: *If you have your own X.25 application that does not use the* `x25config` *program to configure a link, you need to use only the system file installation and kernel configuration portions of* `install.x25`.

6.1. Invoke `install.x25`

As `root`, enter the command:

```
hostname# /usr/sunlink/x25/install.x25
```

then read the instructions displayed by the command and respond to its prompts, which are described below. Note that pressing [Return] accepts the default response offered by the script. `install.x25` displays the default response in square brackets (`[]`). Also note that, in response to any query, you can enter a question mark (`?`) to obtain "help" text.

6.2. Script Introduction

After noting the location of a log file for your inputs (`/usr/tmp/install.x25`), the script outlines the configuration steps and suggests the type of information that you need to complete configuration. If you have obtained the information described in Chapter 3, you can respond to the prompts in `install.x25`.

`install.x25` then queries:

```
Do you have this information and wish to continue? (y/n)[y]:
```

Press [Return] to continue.

If you are using *only* the HDLC service component of SunNet X.25, you answer the initial queries as follows:

```
Do you have this information and wish to continue? (y/n)[y]: return
Do you want to run X.25 over HDLC? (y/n)[y]: n
Do you want to run X.25 over the LAN using LLC2? (y/n)[y]: n
Run X.29? (y/n)[y]: n

  You have no HDLC links hence you will not be able to run IP or OSI over
  X.25.

Do you wish to create /dev/ifd entries? (y/n)[n]: y
```

`install.x25` reports: `Adding entries to /dev`. It then proceeds to the installation of configuration files, as described in Section 6.8.

If you are using X.25, we pick up the script immediately after the `...wish to continue?` prompt.

The following two prompts introduce the HDLC (PSDN) and LLC2 (LAN) branches of the installation script.

```
Do you want to run X.25 over HDLC? (y/n)[y]:
Do you want to run X.25 over the LAN using LLC2? (y/n)[y]:
```

The queries are not mutually exclusive. You can have both PSDN and LAN connections on the same machine.

We now diverge to cover HDLC and LLC2 separately.

6.3. HDLC Configuration

If you pressed Return in response to the X.25 over HDLC prompt above, `install.x25` displays the synchronous serial interface(s) available on your machine.

For each synchronous interface (serial port), `install.x25` follows a sequence in which it asks if you want to run X.25 over that port and then leads you through a set of prompts to create a link configuration file for that port. By convention, link configuration files have names of the form `x25params`*n*, where *n* is a link number, starting with 0 and incrementing with each successive port configuration. You can have one link for each synchronous serial port on your machine.

The script continues with the prompt:

```
Run X.25 over zss0? (y/n)[y]:
```

(`zss0` is an example. `install.x25` starts with the lowest-numbered serial port it finds.) Press Return if you want to run X.25 over the indicated port. Enter `n` to reject the option. If you enter a positive response, `install.x25` begins the process of or creating a new link configuration file for your first link, link 0.

Before building a new link configuration file, `install.x25` tells you to use `/usr/sunlink/x25/edit.x25params` if you need to change any values after exiting `install.x25`.

NOTE: *If you make a mistake (i.e., a valid, but incorrect value) in entering X.25 configuration information, make a note of the mistake, then run* `/usr/sunlink/x25/edit.x25params` *after exiting* `install.x25`*. The* `install.x25` *script actually uses* `edit.x25params` *to perform this portion of script, so that if you later use* `edit.x25params` *you will recognize its user interface. Using* `edit.x25params` *is far less time-consuming than aborting (with a* Ctrl-C*) and restarting* `install.x25`.

The prompts for the X.25 link configuration parameters are as follows:

```
Enter the baud rate for zss0 [9600]:
```

Enter the data rate that you use to connect to your PSDN. Press Return if the rate is 9600 bps.

```
Enter the full X.121 address for link 0 (or "null" if none):
```

Enter the address, including DNIC, NTN, and subaddress, as an unbroken string of decimal digits.

```
How many digits of subaddress in <X.121_address>? [2]:
```

Press Return if you have a two-digit subaddress. Otherwise, enter the number of subaddress digits. The script then displays your X.121 address, with hyphens separating your DNIC, NTN, and (if present) subaddress.

```
Use X.25(1984) on link 0? (y/n)[y]:
```

If your PSDN supports the 1984 X.25 recommendation, press Return. Otherwise, enter `n`.

```
Default throughput (in bits per second)? [9600]:
```

This is the default throughput for you subscribed for when you connected to your PSDN. SunNet X.25 uses the figure entered here when negotiating the throughput for incoming Call Requests.

The next three prompts ask for ranges of logical channel numbers (lcn's) for one-way incoming, two-way, and one-way outgoing calls.

```
Enter the range of incoming lcn or "empty" [empty]:
Enter the range of two-way lcn or "empty" [512 575]:
Enter the range of outgoing lcn or "empty" [empty]:
```

Enter the ranges specified by your PSDN for your particular subscription. Enter a range as two decimal numbers separated by a space, as indicated in the default for two-way channels. If you only have two-way channels (the most common situation), press [Return] in response to the incoming and outgoing prompts. If you do not have two-way channels, enter `empty` for that range.

The next series of prompts relates to how much of the source and destination X.121 addresses will be used in outgoing packets. The conventions of your PSDN determine your responses to these prompts.

```
Exclude the DNIC in the called address for local calls? (y/n)[n]:
The calling address is:
    full:       Send the complete X.121 (DNIC, NTN, and Subaddress).
    no_dnic:    Exclude the DNIC (i.e. send the NTN and the subaddress).
    subaddress_only: Send only the subaddress.
Choose one of the above [full]:
Prepend a prefix digit to calling and called addresses that have a DNIC? (y/n)[n]:
```

If you enter a `y` for the last prompt, you receive the following prompt:

```
Enter the prefix digit to prepend [0]:
```

Press [Return] if your PSDN requires that a `0` be prepended to a source or destination address, or enter the appropriate number.

At this point, if you have more than one synchronous serial port installed on your machine, `install.x25` asks if you want to run X.25 over the next serial port it finds. If you plan to have multiple links, press [Return]. The script then repeats the prompts just described.

When you are finished configuring HDLC links, the script asks if you want to run X.25 over a LAN using LLC2. If you do, see the following subsection; if you do not, enter `n` and proceed to the section, "X.29 Prompts."

6.4. LLC2 Configuration

When HDLC configuration is complete, `install.x25` prompts:

```
Do you want to run X.25 over the LAN using LLC2? (y/n)[y]:
```

If you press [Return], `install.x25` displays the LAN interface(s) available on your machine.

For each interface, `install.x25` follows a sequence in which it asks if you want to run X.25 over that interface, and then leads you through a set of prompts to create a link configuration file for the interface. By convention, link configuration files for LLC2 links have names of the form `x25params`*n*`_llc`, where *n* is a link number, starting with 0 and incrementing with each successive serial port or LAN interface configuration. You can have one link for each synchronous serial port and/or LAN interface on your machine.

Continuing with the script, it prompts:

```
Run X.25 over le0? (y/n)[y]:
```

(`le0` is an example. The script starts with the lowest-numbered interface it finds.)

Press [Return] if you want to run X.25 over the indicated interface. Enter `n` to reject the option. If you enter a positive response, `install.x25` begins the process of creating an `x25params0_llc`, file for your first link, link 0. (If you have already configured an HDLC link, this link number will be greater than zero.)

At this point, `install.x25` tells you to use `/usr/sunlink/x25/edit.x25params` if you need to change any values after exiting the `install.x25` script.

NOTE: *If you make a mistake (i.e., a valid, but incorrect value) in entering X.25 configuration information, make a note of the mistake, then run* `/usr/sunlink/x25/edit.x25params` *after exiting* `install.x25`. *The* `install.x25` *script actually uses* `edit.x25params` *to perform this portion of script, so that if you later use* `edit.x25params` *you will recognize its user interface. Using* `edit.x25params` *is far less time-consuming than aborting (with a [Ctrl-C]) and restarting* `install.x25`.

The prompts for the X.25/LLC2 link configuration parameters are as follows:

```
Enter the full X.121 address for link 0 (or "null" if none):
```

Enter the string `null` if you do not have an X.121 address. If you do have an X.121 address, enter it as an unbroken string of decimal digits, including DNIC, NTN, and subaddress.

If you entered a valid X.121 address at the previous prompt, the script continues:

```
How many digits of subaddress in <X.121_address>? [2]:
```

Press [Return] if you have a two-digit subaddress. Otherwise, enter the number of subaddress digits. The script responds with a display of your X.121 address with hyphens separating your DNIC, NTN, and (if present) subaddress.

```
Default throughput (in bits per second)? [9600]:
```

Default throughput is not used for LAN operation, unless your LAN connects to a PSDN via an X.25 Interworking Unit (IWU). If you have such an IWU connection, obtain the default throughput value from your network administrator.

The next three prompts ask for ranges of logical channel numbers (lcn's) for, respectively, one-way incoming, two-way, and one-way outgoing calls.

```
Enter the range of incoming lcn or "empty" [empty]:
Enter the range of two-way lcn or "empty" [512 575]:
Enter the range of outgoing lcn or "empty" [empty]:
```

Enter the ranges specified by your LAN administrator. Enter a range as two decimal numbers separated by a space, as indicated in the default for two-way channels. If you only have two-way channels (the most common situation), press Return in response to the incoming and outgoing prompts. If you do not have two-way channels, enter `empty` for that range.

At this point, if you have more than one LAN interface installed on your machine, `install.x25` asks if you want to run X.25 over the next interface it finds. If you plan to have multiple LLC2 links, press Return. The script then repeats the prompts just described, starting with the prompt for your X.121 address.

When you are finished configuring LLC links, proceed to the following subsection.

6.5. X.29 Prompts

After you finish with HDLC and/or LLC2 link configuration, `install.x25` presents the following prompt:

```
Run X.29? (y/n)[y]:
```

If you press Return in response to the prompt above, `install.x25` writes a command to start up the `x29` program (X.29 server application) in the X.25 startup file and queries:

```
Allow reverse charge in X.29? (y/n)[y]:
```

If you press Return, `install.x25` adds an argument that allows reverse-charging (for incoming calls) to the just-created `x29` startup command.

6.6. IP over X.25

See Chapters 3 and 4 for considerations when choosing dynamic or static virtual circuits and an explanation of the information you need to respond to `install.x25`'s prompts relating to IP routing.

After the X.29 prompts, if you previously configured one or more HDLC links, `install.x25` continues:

```
Run IP over X.25? (y/n)[y]:
```

If you press Return, the script presents the following menu:

```
    IP over X.25

Select one of
    1. Static IP links.
    2. Dynamic IP links.
    3. DDN dynamic IP links.
    4. None of the above.
 [2]:
```

Press Return if you want to use dynamic IP links. Enter `1` or `3` if you want to use static IP links or connect to the DDN.

SunNet X.25 supports a single IP (including DDN) dynamic-link configuration per machine and multiple static-link configurations per link. A dynamic-link configuration can share a link with one or more static-link configurations.

If you start by configuring static links, `install.x25` lets you set up multiple static configurations per physical link, for each of the physical links you just configured. When static configuration is complete, the script asks if you want to configure a dynamic IP link. It allows a single dynamic-link configuration.

If you start by configuring dynamic links, `install.x25` lets you set up a single dynamic-link configuration, then begins the sequence of prompts for multiple static configurations described in the preceding paragraph.

IP Dynamic

If you select dynamic IP links, you receive the following prompt:

```
Enter the IP address for the IP dynamic interface over X.25:
```

Enter a valid hostname or an IP address. A valid hostname is one that is listed in your `/etc/hosts` file or in your network information service's `hosts` database. If you enter a numeric address, use the Internet "dot" notation.

Note that for the IP dynamic interface, the IP address (or equivalent hostname) you enter at the prompt above must have a network number different from the IP address of the primary network interface on your machine. On most machines, the primary network interface is the Ethernet or FDDI interface.

Next, the script prompts you to enter the netmask used in your network, offering `none` as a default. If you use a netmask, enter a plus sign (`+`). This causes `install.x25` to obtain your netmask value from your local `/etc/netmasks` file or from the network information service's `netmasks.byaddr` database. If you enter a numeric netmask value, use the Internet "dot" notation.

After entering your IP address and netmask, you receive the following sequence of prompts:

```
Enter a destination IP address reachable over X.25: <hostname>
Enter the corresponding full destination X.121 address: nnnnnnnnnn
Enter another destination? (y/n)[y]:
```

Enter a valid hostname or numeric IP address followed by the X.121 address that corresponds to that IP address. Press Return if you want to enter another destination† or enter `n`. If you enter `n`, the script asks if you want to run IP static over the same link. If you enter `y`, the script redisplays the IP-over-X.25 menu of link types. Enter `1` to select static IP links.

DDN Dynamic

From the IP-over-X.25 menu, if you select `3`, to configure DDN dynamic virtual circuits, you receive:

```
Enter the IP address for the IP DDN interface over X.25:
```

Enter a valid hostname or IP address. A valid hostname is one that is listed in your `/etc/hosts` file or in your network information service's `hosts.byname` database. If you enter a numeric address, use the Internet "dot" notation.

Note that for the DDN dynamic interface, the IP address (or equivalent hostname) you enter at the prompt above must have a network number different from the IP address of the primary network interface on your machine. On most machines, the primary network interface is the Ethernet or FDDI interface.

Next, the script prompts you to enter the netmask used in your network, offering `none` as a default. If you use a netmask, enter a plus sign (`+`). This causes `install.x25` to obtain your netmask value from your local `/etc/netmasks` file or from the network information service's `netmasks.byaddr` database. If you enter a numeric netmask value, use the Internet "dot" notation.

After this, you receive:

```
Enter the number of the DDN link [0]:
```

If you want to run DDN over a link different from 0, enter the link number here.

After this, the script asks if you want to run IP static over X.25. If you enter `y`, you return to the IP-over-X.25 menu.

---

† The X.25 dynamic virtual circuit feature (for both DDN and IP) provides a point-to-multipoint interface, so that you can have multiple paths to different destinations over the same X.25 link.

IP Static

After selecting `1` from the IP-over-X.25 menu, you receive the prompt:

```
Enter the IP address for the IP static interface over hdlc0 [maestro]:
```

The script offers the hostname of your machine (`maestro` in the example above) as a default. Press Return to accept the default, or enter a valid hostname or IP address. A valid hostname is one that is listed in your `/etc/hosts` file or in your network information service's `hosts.byname` database. If you enter a numeric address, use the Internet "dot" notation.

Next, the script prompts you to enter the netmask used in your network, offering `none` as a default. If you use a netmask, enter a plus sign (`+`). This causes `install.x25` to obtain your netmask value from your local `/etc/netmasks` file or from the network information service's `netmasks.byaddr` database. If you enter a numeric netmask value, use the Internet "dot" notation.

After entering your IP address, you receive the following sequence of prompts:

```
Enter a destination IP address reachable over X.25: <hostname>
Enter the corresponding full destination X.121 address: nnnnnnnnnn
```

Enter a hostname or numeric IP address followed by the X.121 address that corresponds to that IP address.

The next prompts are:

```
Select mode: caller (y), or called (n) (y/n): n
Is this a permanent VC? (y/n)[n]:
```

Note that, over the same link, you can be "caller" for one configuration and "called" for another.

6.7. Installing `ifd` Devices

The `ifd` devices referred to in the following prompt are required if you plan to run an application directly on the HDLC interface, instead of over X.25 and HDLC. They are also required by the `syncloop` troubleshooting utility, described in Chapter 9.

```
Do you wish to create /dev/ifd entries? (y/n)[n]:
```

If you enter `y`, the script reports: `Adding entries to /dev`.

When you are finished with link configuration, the script prompts:

```
What release of SunOS are you running (e.g. 4.0.3, 4.1)? [4.1]:
```

The script performs an internal check on the SunOS version you are running, then displays as a default what it "thinks" your SunOS version is. Press Return to confirm the default or enter the correct version number. If you are running SunOS version 4.1, the script then displays: `Installing a wrapper for x29`. A wrapper is software that is added to the `x29` program to ensure POSIX conformance.

6.8. Installing X.25 Configuration Files

After you enter your SunOS version number, `install.x25` begins the process of installing or updating your X.25 configuration files. You receive the following series of prompts:

```
Install (or update) /etc/sunlink/x25/rc? (y/n)[y]: <return>

Install (or update) /etc/sunlink/x25/x25manager.configuration? (y/n)[y]: <return>

Install (or update) /etc/sunlink/x25/x25params0? (y/n)[y]: <return>

Install (or update) /etc/sunlink/x25/x25params1? (y/n)[y]: <return>

Install (or update) /etc/sunlink/x25/x25params2_llc? (y/n)[y]: <return>

Install (or update) /etc/sunlink/x25/x29-auth.sample? (y/n)[y]: n

Install (or update) /etc/sunlink/x25/x29profile.sample? (y/n)[y]: n

Install (or update) /etc/sunlink/x25/routes.sample? (y/n)[y]: <return>

    Modifying /etc/rc.local to call /etc/sunlink/x25/rc.
    Saving original /etc/rc.local in /etc/rc.local.nnnn...
```

It is important that you install the `/etc/sunlink/x25/rc` file and an `x25params` file for each of your links. You need the `x25manager.configuration` file if you run IP (including DDN) or CLNP over X.25. You need the `x29-auth.sample` and `x29profile.sample` files only if you plan to use the X.29 server and PAD programs that are shipped with SunNet X.25. The `routes.sample` file gives examples of how you can route calls over multiple links. This feature is described in Section G.

If you do not want to start X.25 each time you reboot your machine, then edit your `rc.local` file (as `root`) to comment out the lines, shown below, that start the X.25 functions.

```
#
# Start X.25
#
if [ -f /etc/sunlink/x25/rc ]; then
   /etc/sunlink/x25/rc
fi
#
```

6.9. Install System Files

After installation of configuration files, the next prompt you see is:

```
Install (or update) your system files? (y/n)[y]:
```

These files are required for the correct operation of SunNet X.25. Press [Return] to start the process of system-file installation. The script begins installing system files, saving existing files as it does so. In the following order, the script installs:

- X.25 kernel object files
- X.25 kernel source files
- X.25 include files

At this point, you are ready to configure a new kernel.

6.10. Kernel Configuration

The kernel-configuration portion of the script begins:

```
Would you like to have a kernel configured? (y/n)[y]:
```

If you are running `install.x25` for the first time for the current release, press [Return].

NOTE: *If you re-run* `install.x25` *to change the number of HDLC or LLC2 links or the number of static or dynamic IP links, you must reconfigure your kernel.*

The script prompts you:

```
Enter the name of an existing kernel configuration file [SDST60]:
```

The script supplies as a default the name of the configuration file for the running kernel. If you are running a `GENERIC` kernel, then before accepting `GENERIC` as your baseline for a new kernel, read section 3.4 about size considerations for GENERIC kernels.

`install.x25` then prompts:

```
Enter the name of the new kernel configuration file [X25]:
```

Press [Return] to accept `X25` as the name of your new kernel configuration file, or enter a name of your choosing. If the specified file already exists, the script asks your permission before overwriting it. The default option is not to overwrite. Enter `y` to allow overwriting.

`install.x25` displays the full pathname of the log file, in the directory `/etc/sunlink/x25/kernel`. The log file name is the name of your kernel configuration file with the extension `.log`. After displaying the log file location, the script reports that it saved your old `/usr/sys/`*arch_k*`/conf/files` (*arch_k* is your kernel architecture) file and is using an updated version of that file.

If you have X.25 options in your kernel, either from a previous release or previous installation of SunNet X.25, the script asks your permission before removing the old options and entering the new. The script then continues:

```
Would you like to have a new kernel built
(this will take some time)? (y/n)[y]:
```

Press Return and you receive the prompt:

```
Would you like this script to install the kernel and *reboot* after
a successful kernel build? (y/n)[y]:
```

If you enter `n` to this prompt, you must “manually” save your existing `/vmunix`, move the new `vmunix` to `/`, and reboot. Whether you select automatic reboot or not, the script runs `config` on your modified kernel configuration file. When `config` completes, the process of compiling and loading your new kernel begins. When compiling and loading is complete, then if you accepted the automatic-reboot option, the script saves your `/vmunix` as `/vmunix.old`, moves the `vmunix` from your configuration directory to `/vmunix`, and invokes `fastboot`.

If your kernel does not compile and load successfully, check the log file in `/etc/sunlink/`*prod_name*`/kernel/`*config_filename*`.log` (where *prod_name* is `osi` or `x25`) for any obvious problems. If the log file is unenlightening, use the instructions in the SunOS manual, *System and Network Administration*, to build a `GENERIC` kernel, i.e., one without OSI and X.25 devices. If your attempts to build an OSI/X.25 kernel fail, contact your local Sun Answer Center and inform the person there whether the failure was confined to an OSI/X.25 kernel or whether you were also unable to build a `GENERIC` kernel.

If you did not accept the automatic reboot option and the new kernel compiles and loads successfully, you are ready to rename your original kernel, copy the new kernel to root, and boot the new kernel. To do so, enter the following commands:

```
hostname# cd /sys/‘arch -k‘/kernel_config_directory
hostname# mv /vmunix /vmunix.old
hostname# mv vmunix /vmunix
hostname# /etc/reboot
```

where `arch -k` returns your kernel architecture (e.g., `sun4c` for a SPARCstation-1). Replace *kernel_config_directory* with the name of your kernel configuration file.

If the new kernel does not boot successfully, abort your machine (L1- [or Stop] a) and enter the following command to reboot the old kernel:

```
> b /vmunix.old -s
```

The system displays boot-time messages. When you receive the system prompt, enter the commands:

```
# cd /
# mv vmunix.old vmunix
# ^D
```

The last command in the box above is a Ctrl-D. If the new kernel did not boot successfully, retrace your steps and try to determine where a mistake was made. Refer to the section “Kernel Configuration Procedures” in *Installing the SunOS* for guidance. As a further step, use the instructions on kernel configuration in the SunOS manual, *System and Network Administration*, to build a `GENERIC` kernel, i.e., one without OSI and X.25 devices. Then, attempt to boot the `GENERIC` kernel. It is useful for the Sun Answer Center engineer to know whether your booting problem is confined to an OSI/X.25 kernel or extends to `GENERIC` kernels as well.

6.11. The Next Step

If you have a configuration that differs from a standard connection to a PSDN or LAN, you might consult Chapter 7 for an explanation of the commands used to start SunNet X.25.

If you plan to use the PAD or X.29 applications that are shipped with SunNet X.25, see the *SunNet X.25 Application Guide* for instructions on configuring those applications.

7

Starting SunNet X.25

This chapter tells you how to start and stop SunNet X.25. It briefly describes the startup scripts used to initialize and start SunNet X.25. For most of you, the material here will be for reference. This is because the script `install.x25` creates a script that starts SunNet X.25 at boot time.

If you do not want to start SunNet X.25 software at boot time, then comment out the lines (shown below) in `/etc/rc.local` that invoke the startup script.

```
#
# Start X.25
#
if [ -f /etc/sunlink/x25/rc ]; then
       /etc/sunlink/x25/rc
fi
```

If you want to invoke the startup files at run- rather than boot-time, you can, as `root`, invoke the `rc` files as you would any SunOS command. For example:

```
hostname# /etc/sunlink/x25/rc
```

7.1. Commands Used in SunNet X.25 Startup File

SunNet X.25 is started by the file `/etc/sunlink/x25/rc`, which is created by `install.x25`. The `x25/rc` file contains the following commands:

`syncinit` <*portname*> `speed=`<*speed*>
: There is a `syncinit` command for each serial port over which you run X.25. If you are directly connecting two X.25 hosts with a null modem cable, you must add `txc` and `rxc` options to the `syncinit` command(s). See Appendix F for details.

`x25config` <*link_config_file*>
: There is an `x25config` command for each of your links. You use the `/usr/sunlink/x25/edit.x25params` script to make changes to your link configuration file after you finish with `install.x25`. The syntax for `x25config` is described in Section B.5.

`x25mgr` <*x25mgr_config_file*>
: The `x25mgr` program, stored in `/etc/sunlink/x25`, manages virtual

circuits for X.25 when it is used to connect TCP/IP networks (including the DDN) and when you run the CLNP portion of SunNet OSI over SunNet X.25. The single `x25mgr` configuration file can contain sections for each of the possible users (IP, DDN X.25 Standard Service, or CLNP) of the `x25mgr` program. Appendix D explains the syntax for the `x25mgr` program and its configuration file.

`x29` *[*`-r`*]*
If you use the X.29 server application that is shipped with SunNet X.25, `install.x25` adds an `x29` command line, with an optional reverse-charging argument (`-r`). The X.29 server application is documented in the *SunNet X.25 Application Guide*.

7.2. Stopping SunNet X.25

To bring down an X.25 link, from the packet level down, you use the `x25config` program, specifying the `-off` argument and either the link number or the link configuration file for the link you want to bring down. For example:

```
hostname# cd /etc/sunlink/x25
hostname# /usr/sunlink/x25/x25config -off x25params0
```

The command above brings down link 0, the link specified in the link configuration file `x25params0`. We changed directories to `/etc/sunlink/x25` so that `x25config` could find the `x25params0` file.

A command equivalent to the preceding is:

```
hostname# /usr/sunlink/x25/x25config -l0 -off
```

You bring down an LLC2 link in the same way that you bring down an HDLC link. For example, to bring down an LLC2 link on a machine whose only X.25 link is an LLC2 link, enter a command such as the following:

```
hostname# cd /etc/sunlink/x25
hostname# /usr/sunlink/x25/x25config -off x25params0_llc
```

After bringing down a link (over either LLC2 or HDLC), you can restart the link by reinvoking the SunNet X.25 startup file, `/etc/sunlink/x25/rc`.

8

Configuring and Running as an HDLC Service

This chapter describes how to configure and run of SunNet X.25 as an HDLC service. Using the procedures described here, standard SunOS user-level applications can communicate over an HDLC connection. Before starting the steps in this chapter, make sure you have completed the procedures in the Chapter 6 to make `ifd` device entries in `/dev`.

You must run the commands described here each time you reboot. For this reason, we suggest that you add appropriate commands to your `/etc/rc.local` or incorporate them in a shell script.

8.1. Initializing and Layering

In the following example commands, we use `ifd0`. `ifd` devices are numbered from 0 to 15 (maximum of 16 devices). The SunNet X.25 installation script, `install.x25`, creates 16 entries.

As a first step, you must initialize the `ifd` device. To do so, enter the following command:

```
hostname% cat /dev/null > /dev/ifd0
```

Next, layer the `ifd` device to the `hdlc` and serial port devices. To do so, for CPU or MCP ports, enter a command such as the following:

```
hostname# iflayer ifd0 hdlc0 zss0
```

`zss0` refers to the serial port driver for CPU port A on a Sun-3/-4 or SPARCstation. Substitute the name of an MCP port driver, such as `mcph0`, if you are using an MCP port.

8.2. Set HDLC Parameters

Use the `hdlcconf` command to set your machine's HDLC parameters. The first argument to `hdlcconf` is the device to configure. For our example, we use `/dev/ifd0`. To accept the defaults (shown in Table 8-1, below), enter the following command:

```
hostname# hdlcconf /dev/ifd0
```

The device name is the only required argument to the command. Remaining arguments can be in any order.

NOTE: *If you have a "straight cable" connection for your HDLC link, that is, are using a null modem cable or modem eliminator, you must set one side to be the DTE and the other to be the DCE. You do this with the* `dte` *and* `dce` *command-line arguments to* `hdlcconf`*, which are described below.*

The following table specifies the options to `hdlcconf` and lists the defaults for the command.

Table 8-1 `hdlcconf` *Options and Defaults*

Option	Description	Default
`dce/dte`	Specifies the address of the HDLC line.	`dte`
`-w`	Specifies the number of unacknowledged frames that are allowed to be outstanding before the local host must wait for an acknowledgement from the remote host. If window size is greater than 7, you must use `-e` parameter, below, to set the link to operate in HDLC Extended Control Mode. `t1`, `t2`, and `t3` (following) are *rounded up* to the next highest integral multiple of the clock resolution (`tick`).	7
`-tick`	Resolution of the clock used (milliseconds)	`500`
`-t1`	Denotes the time allowed before a retransmission is attempted.	`1500`
`-t2`	Causes the implementation to wait `t2` milliseconds before sending an acknowledgement (RR) to an Information (I) command.	`0`
`-t3`	If the line remains idle for a period exceeding `t3`, the protocol code will poll the remote host (if the link is up) with an RR.	`5000`
`-n1`	Specifies the maximum frame size.	`512`
`-n2`	Sets the maximum number of retries (on a poll frame).	`10`
`-e`	When present, sets the link to operate in HDLC Extended Control Mode. You must use Extended Control Mode if you have a window size (`-w` parameter) greater than 7.	
`-start`	Enables HDLC.	
`-stop`	Disables HDLC.	

If you do not specify the `-start` and `-stop` options, you must use the `hdlcstart` and `hdlcstop` commands, described below, to establish or shutdown the link.

For proper operation of HDLC, it is very important to set the values of `t1`, `t2`, and `t3` correctly. To get optimal performance, you must empirically tune these numbers.

8.3. Initialize Serial Port

Use `syncinit` to initialize the serial port you intend to use. The following command initializes the third port on the first MCP board to run at 9.6 Kb/sec.:

```
hostname# syncinit mcph2 9600
```

8.4. Establish Link-level Connection

With the serial port initialized, use `hdlcstart` to establish a link-level connection. The command accepts an `ifd` device as its only argument. For example:

```
hostname% hdlcstart /dev/ifd0
```

After you run this command, the HDLC layer in SunNet X.25 resets its counters and sends an HDLC command to establish a connection with a host.

8.5. Stop HDLC Communication

Use `hdlcstop` to break the HDLC connection at the local host. The command turns off packet transmission and causes incomplete packets to be discarded. The syntax is identical to that of `hdlcstart`:

```
hostname% hdlcstop /dev/ifd0
```

After you run this command, the HDLC layer in SunNet X.25 resets its counters and flushes all output queues.

8.6. Example of HDLC Use

A SunOS program, such as `dd`, that can access an `ifd` device can use the HDLC service described in this chapter.

After performing the steps described above, you can enter the following `dd` command to copy the contents of the file `myfile` through the serial port layered to `ifd0`:

```
hostname% dd if=somefile of=/dev/ifd0 obs=256
```

`if` names the input file, `myfile`; `of` names the output file, `/dev/ifd0`; and `obs` specifies the output block size, 256, in bytes.

9

X.25 Troubleshooting

This chapter details troubleshooting procedures that you might need should problems arise during the operation of SunNet X.25.

If you are experiencing problems while using SunNet X.25—ranging from abnormally long response times to disconnections— try to determine in which layer the problem originates. Use the `vcstat` command (see Section 9.6) to check whether the link and packet layers are up. Use `x25trace` (see Section 9.7) to observe connection attempts and obtain clearing causes and diagnostic codes. The `syncstat` (Section 9.4) and `netstat (8C)` commands give you error counts for various layers of networking software. These counts provide clues as to the origin of a problem. With information from one of these diagnostic tools in hand, you can find the appropriate heading in the troubleshooting checklist and proceed with the specified steps.

9.1. Troubleshooting Checklist

If you cannot bring up SunNet X.25...

1. If you receive a message such as `Cannot connect to socket`, it probably means that you made a mistake in running `install.x25`. For example, you might have configured only one HDLC link, with the result that `install.x25` would configure only a single set of HDLC devices in your kernel. If you then attempted to run a second X.25 link you would have no support for such a link in your kernel. Retrace your steps in `install.x25` to determine where you might have erred.

   You might also receive `Cannot connect to socket` if X.25 software is not present in the kernel.

2. After trying to start SunNet X.25 for the first time, if you receive the message `WR_SYSGEN_TIMED OUT`, it might indicate that your cable and modem are incorrectly matched to your machine. Another possibility is that you specified the wrong serial port when you ran `install.x25`. Check your `/etc/sunlink/x25/rc` file to make sure that the `syncinit` and `ifconfig` commands specify the serial port(s) (`zss`<*n*>, `mcph`<*n*>, or `hss`<*n*>) over which you plan to run SunNet X.25. If your port specifications in `/etc/sunlink/x25/rc` are correct, try replacing the cable between the serial port and modem. If replacing the cable does not correct the problem, you might need to use a different modem.

If you cannot bring up or maintain the X.25 link layer (HDLC or LLC2)...

1. A message such as: `X25_WR_SYSGEN for linkid` *n* `Network is down` indicates the layering of protocol devices, as performed by the `iflayer` command(s) in your `/etc/sunlink/x25/rc` startup file, has not occurred. Try re-running `rc`.
2. Make sure all cables are properly seated.
3. While you still have a connection, check the Transmit and Receive Data lights on your modem: with HDLC transmission, these should always be on.
4. If you have a connection, use a breakout box at the Sun (not the modem) end of the connection to check that (RS-232-C) pins 2, 3, 4, 5, 6, 7, 8, 15, 17, and 20 are functioning. Pay particular attention to the Transmit Clocking and Receive Clocking pins, 15 and 17. If the lights corresponding to these pins are not lit, it indicates that the local and/or remote modems are not supplying clocking. Notify the remote end and use your modem's documentation to perform troubleshooting.
5. If you are operating over an HDLC link, make sure that you are using a cable that is designed for synchronous use. If you have a spare RS-232-C, RS-449, or V.35 cable, replace your existing cable and retry the connection.
6. Contact your PSDN (or LAN administrator for LLC2 links) to verify that your parameter settings are correct. In particular, for HDLC links, check that the link is configured as a DTE if communicating over a PSDN. If you discover errors, use `/usr/sunlink/x25/edit.x25params` to make changes to your link configuration file.
7. If you are using a null modem connection, one side should be a DTE, the other a DCE and one or both sides must supply clocking. See Appendix F for instructions on setting up a null modem connection.
8. If you are confident that your parameter settings match those of the PSDN, use the `syncstat` command to observe the number of packets being sent and received over a port and whether errors are being detected at the hardware or port driver level. If `syncstat` reveals errors, use `syncloop` to perform a loopback test. You should at least loopback through your local modem, to verify the integrity of the link from your machine through the modem. If you use `syncloop` to test your link up through the local modem (option 1), be sure to use `syncinit` afterwards to take the link out of loopback mode. The `syncstat` command is described in Section 9.4; `syncloop` is described in section 9.3. You can use `syncloop` and `syncstat` with any of Sun's serial port options.
9. If you are running SunNet OSI over SunNet X.25, use `/usr/sunlink/osi/etc/x25trace` to check the HDLC or LLC2 level. Use a command such as the following:

```
hostname# x25trace -i hdlc0 hdlc
```

For an LLC2 link, use:

```
hostname# x25trace -i llc0 llc
```

The x25trace command is described in Section 9.7.

If you are unable to establish a packet-level connection (virtual circuit)...

1. Try to evaluate the call clearing code returned from the PSDN. See Tables 9-4 and 9-5. for a complete list of clearing codes and possible causes.

2. If you receive an error code of the form 00 *nn*, the leading zeroes indicate the call-clearing request originated with the remote host. If the first two digits are not 00, it indicates the call-clearing request originated in the PSDN. See Section 9.8 for a complete list of error codes that can be returned from the PSDN or the remote host.

3. Two commonly received error codes are 0x00 24 and 0x00 43. If you receive an error code of 0x00 24, check that the logical channel number ranges specified in your x25params file are within the range specified by your PSDN. If you receive an error code of 0x00 43, check that your X.121 address is specified correctly in your x25params file. If it is, check to see whether there is a process (such as x29, if you are using pad) listening for an incoming call.

4. If you receive error code 0x00 42, it is likely that your maximum I-frame size (max_frame_size), an HDLC-level parameter, is set incorrectly for your link. The maximum I-frame size is a subscription option to the PSDN. You must set the max_frame_size parameter to be eight† bytes larger than the default packet size parameter (def_packet_size), at the packet level. If SunNet X.25 receives a Call Request with a packet size that is within eight bytes of the max_frame_size parameter, the X.25 software clears the call with the 0x00 42 diagnostic. Use /usr/sunlink/x25/edit.x25params to modify x25params as necessary.

5. If you can find out about the remote host, check that it is still up.

6. Run ifconfig commands to check on the hdlc and xpkt protocol devices. (Use of ifconfig is described in section 9.5.) If the response from ifconfig is UP, POINTOPOINT, and RUNNING, check with your PSDN to see if there is a problem at the network end.

7. Run x25trace to trace the exchange of X.25 packets across the link. People who are familiar with the X.25 Packet Layer Protocol can interpret the output from x25trace to determine where errors are occurring. The x25trace command is described in Section 9.7.

† The value for max_frame_size should be eight bytes larger than the default packet size when HDLC is in extended mode and five bytes larger when HDLC is in normal mode.

If the link is up, but not responding correctly

1. If you experience intermittent disconnections or resets, check for configuration errors in your link configuration file. At the packet level, make sure that the window and packet size parameters agree, between your machine and the PSDN.

2. Check console messages. These occasionally indicate the source of a problem. If you receive console messages such as

   HDLC_FRMOUT[<*link_num*>]: CTL=??, Ns=?, Nr=?, RESP=0, why = Y

   or

   HDLC_FRMIN[<*link_num*>]: CTL=??, Ns=?, Nr=?, RESP=0, why = Y

   it indicates that a frame was received (first message) or sent (second) that was larger than expected by the Sun (for the first message) or the PSDN (for the second). Such messages indicate that, at the HDLC level, the setting for the `max_frame_size` parameter is not in agreement with the PSDN's setting for your link. The value for `max_frame_size` should be at least eight bytes larger than the size of the maximum X.25 layer packet to which you subscribed.

3. If you experience poor response times using SunNet X.25, it might be because you are communicating over a series of physical links, one or more of which might be shared with other users. Your throughput will be limited by the slowest link among the links that make up your connection to the remote host.

4. If you establish but cannot maintain a connection and none of the preceding steps uncovers your problem, run `x25trace`, as described in section 9.7. Output from this command is valuable in helping trained technical personnel find and correct problems.

If you receive boot-time messages that delay the booting process

You might receive console messages such as:

```
zs0: silo overflow
zs0: parity error ignored
zs0: parity error ignored
zs0: parity error ignored
```

where `zs0` is the name of the driver for CPU port A. These messages do not indicate a problem, however, they can delay the booting process. To suppress these messages, as `root`, edit the file `/etc/ttytab`. In this file, find the lines:

```
ttya   "/usr/etc/getty std.9600"   unknown on   local secure
ttyb   "/usr/etc/getty std.9600"   unknown on   local secure
```

The device `ttya` is associated with CPU port A; `ttyb` is associated with CPU port B. Comment out (with a hash mark [#]) the line corresponding to the port(s) over which you run SunNet X.25.

9.2. Software Port Names

The following port naming conventions are used by initialization and serial port diagnostic commands.

For a Sun-3/-4 (including SPARCstations and SPARCservers) software port names for CPU ports are of the form `zss`<*n*>, where <*n*> is a number in a range starting with 0 and ending at one fewer than the number of serial ports on your machine. For example, on a SPARCstation-1+ with two serial ports, the serial ports are named `zss0` and `zss1`, referring, respectively, to hardware ports A and B.

Most CPU ports are RS-232-C-compatible. However, you should check your hardware documentation to confirm that your machine's CPU ports support SunLink and SunNet products.

Names for MCP ports are of the form `mcph#`, where # is a number in the range from 0-15. Ports on the first MCP board are designated `mcph0` through `mcph3`; ports on the second board are `mcph4` through `mcph7`. The naming scheme is the same for ports on a third and fourth board, if present. On an MCP board, ports 0 and 1 are RS-449; ports 2 and 3 are RS-232-C.

HSI port names have the format: `hss`*y*, where *y* represents the port number, `0` or `1` for the first HSI board and `2` or `3` for the second. For example, `hss1` is the name for port 1 (the second port) on the first HSI board. `hss3` is the name for port 1 on the second HSI board.

The software port numbers, as used in HSI port names, have the following relation to the hardware port numbers used on each board:

Table 9-1 *HSI Hardware and Software Port Numbers*

Board No.	Hardware Port No.	Software Port No.
1	0	0
	1	1
2	0	2
	1	3

NOTE: *In order to use the* `syncloop` *command, described below, you must have installed* `ifd` *devices in* `install.x25`*, as described in Section 6.7.*

9.3. Loopback Testing with `syncloop`

`syncloop` is shipped with the SunLink HSI and SunLink MCP products. If you are using either of these products, refer to the `syncloop` documentation that accompanies your product. The `syncloop` description here is intended for those who are using a CPU serial port for their X.25 connection and do not have `syncloop` documentation.

The `syncloop` command performs a loopback test that checks the following components of your communications link:

- your port-driver software layering
- CPU-to-port communication
- the correct operation of the serial port
- the cable from port to modem (or modem equivalent)
- the local and remote modems (or modem equivalents)
- the transmission line

To run `syncloop,` you must have `/usr/sunlink/sync` in your path.

When you invoke `syncloop`, it transparently invokes the `iflayer` command to connect an `ifd` device to a specified serial port. In most cases, it then runs the `syncinit` command to initialize the serial port and sends out packets. `syncloop` then reads incoming packets to verify that they were received.

Do not run `syncloop` on a port that is in use. Because certain `syncloop` options put a port in loopback mode, its use can prevent communication with a remote host during the time that `syncloop` is sending and receiving packets. To quiesce an active port (in this example, `zss0`) enter the following `syncinit` command:

```
hostname# syncinit zss0 0
```

After you finish with `syncloop`, you must re-run `/etc/sunlink/x25/rc`, to reinitialize your serial port.

A `syncloop` command takes the following general form:

```
hostname# syncloop [options] portname
```

where *portname* is in the format described in section 9.2.

The options for `syncloop` are described in the following table.

Table 9-2 *Options to the* `syncloop` *Command*

Option	Parameter	Description
`–i`	`ifd_number`	Specifies which `ifd` device to use; the default device is `/dev/ifd0` (`–i 0`).
`–c`	`packet_count`	Specifies the number of packets used for data transfer. The default is 100.
`–l`	`packet_length`	Specifies the length of the packet in bytes. The default is 100, and the maximum packet length the user can specify is 1152.
`–s`	`line_speed`	Bit rate in bits/sec.

Table 9-2 *Options to the* `syncloop` *Command— Continued*

Option	Parameter	Description
–t	test_type	A value in the range 1-4 that specifies the test_type option (see descriptions of test_type options in following subsection).
–d	hex_data_byte	Specifies a hexadecimal number as the byte content of each packet. The default is to use random data.
–v		Sets verbose mode. If data errors occur, the expected and received data are displayed.

Enter all numeric options except `hex_data_byte` as decimal numbers (for example, `-t 3`). If you do not provide the `test_type` option, `syncloop` prompts for it, as in the following example:

```
hostname# syncloop zss1
[Using /dev/ifd0 ]
[iflayer ifd0 zss1 ]
Enter test type:
1: Internal Test
2: Test using loopback plugs
3: Test using local or remote modem loopback
4: Other, previously set, special mode
> 1
```

The following table lists the equipment required for each `test_type` option:

Table 9-3 *Equipment Needed for* `test_type` *Options*

Option Number	Equipment
1	none needed
2	loopback plug
3	modem(s)
4	uses previous setup

`test_type` Options

The following are descriptions of the available `test_type` options:

Test Option 1 — Internal Test

This option uses the internal clocking and internal loopback and runs the following `syncinit` command:

```
syncinit portname speed loopback=yes txc=baud rxc=baud
```

The test data packet will be sent to the specified serial port and looped back internally.

NOTE: *When you are finished with the loopback test, run* `syncinit` *specifying the same serial port you specified in your* `syncloop` *command, and enter* `loopback=no` *to take the port out of loopback mode.*

Test Option 2 — Test Using Loopback Plugs

This option uses the internal clocking and runs the following `syncinit` command:

```
syncinit portname speed loopback=no txc=baud rxc=rxc
```

This test data packet will loop between the CPU and serial port through the loopback plug. Before using this option, install the appropriate loopback plug (RS-232-C for CPU serial ports) on the specified port.

Test Option 3 — Test Using Local or Remote Modem Loopback

This option uses the external clocking set by the modem and runs the following `syncinit` command:

```
syncinit portname speed loopback=no txc=txc rxc=rxc
```

Testing with a modem in local loopback mode verifies proper operation of the serial port, the external cable, and the local modem. Testing with the modem in remote loopback mode, checks the components just mentioned, as well as verifying the operation of the communications link.

With this `test_type` option, `syncloop` treats both local and remote modem loopback testing the same, since clocking is provided by the modem in both cases. Whether the data is looped back through the local or remote modem depends on how the modems are set up.

If the test fails on remote modem loopback, but succeeds on local modem loopback, carefully check the transmission line and the setup of both modems.

Test Option 4 — Use Previously Set Mode

There is no automatic `syncinit` execution with this option. This allows you to use `syncinit` before running `syncloop` to specify clocking and loopback options for the port that are not possible with the other `syncloop` test options.

Specifying `ifd` Devices

`syncloop`, like several SunLink and SunNet products, uses an `ifd` device to connect to a serial port driver. A specific `ifd` device cannot be shared. If you run `syncloop` twice at the same time or try to use `syncloop` while another SunLink or SunNet product is running on the same machine, you might receive a `device busy` message. If this occurs, use the `netstat -i -a` command to determine which `ifd` device is in use. Then, use `syncloop`'s `-i` option, described above, to specify a different `ifd` device from the one in use. `ifd` device names have the form `ifd`# where # is a number in the range from 0 to 15.

`syncloop` Output

When the loopback test runs successfully, using any of the `test_type` options, `syncloop` reports the statistics shown in the following sample output and then terminates.

```
hostname# syncloop zss1
[ Using /dev/ifd0 ]
[ iflayer ifd0 zss1 ]
Enter test type:
1: Internal Test
    (internal data loop, internal clocking)
2: Test using loopback plugs
    (external data loop, internal clocking)
3: Test using local or remote modem loopback
    (external data loop, external clocking)
4: Other, previously set, special mode
> 1
speed=9600, loopback=yes, nrzi=no, txc=baud, rxc=baud
[ syncinit zss1 9600 loopback=yes txc=baud rxc=baud ]
[ checking for quiet line ]
[ Trying first packet ]
[ Trying many packets ]
 100
100 packets sent, 100 received
0 CRC errors, 0 aborts, 0 overruns, 0 underruns
estimated line speed = 9213 bps
```

9.4. Monitoring the Link with `syncstat`

The `syncstat` command gives you information about the packets transmitted and received on a synchronous serial line. It is a valuable tool for monitoring your X.25 link.

The syntax for `syncstat` is:

```
# syncstat port_dev [interval]
```

You must be `root` to run the command. The first argument in a `syncstat` command is the name of the serial port (see section 9.2 for a description of port name syntax). If you enter the command with no additional arguments, `syncstat` reports cumulative statistics since system boot time. For example:

```
hostname# syncstat mcph1
 speed  ipkts   opkts  undrun  ovrrun   abort   crc   isize   osize
  9600  15716   17121       0       0       1     3      98      89
```

where:

`speed` Reports the line speed as set by `syncinit`. It is the system administrator's responsibility to make this value correspond to the modem clocking speed when clocking is provided by the modem.

`ipkts` Reports the total number of input packets.

`opkts` Reports the total number of output packets.

`undrun` Reports the number of transmitter underrun errors. Such errors occur when the local system is too busy to service the serial port hardware. A frame that is not completely sent is aborted, triggering error recovery. Underrun errors can occur when the signaling rate in use on a link is too fast for the local system.

`ovrrun` Reports the number of receiver overrun errors. Such errors occur when the local system is unable to accept data fast enough and the port hardware buffers overflow. A frame that is not completely received is aborted, triggering error recovery. Underrun errors can occur when the signaling rate in use on a link is too fast for the local system.

`abort` Reports the number of aborted received frames. Occurs when the local serial port received a sequence of eight consecutive ones, in violation of HDLC framing rules. Abort errors result from an interruption in the service provided by the link or from clocking problems. Such errors might also be caused by the software running over SunNet OSI. A small number of abort errors probably indicates a software problem rather than a broken link or a persistent clocking problem.

`crc` Reports the number of received frames with CRC (Cyclical Redundancy Check, an error detection method) errors. A CRC error is recorded when the checksum on a received frame is incorrect. CRC errors occur when there is a clocking problem (different rates on each side) or a noisy line.

`isize` Reports the average size of input packets.

`osize` Reports the average size of output packets.

Errors under `undrun`, `ovrrun`, `abort`, and `crc` might indicate a problem with your serial port hardware, connectors, cables, or line-interfacing equipment. If you experience such errors, use `syncloop` (or an equivalent loopback diagnostic) to determine which component of your physical link is causing the errors.

If you do not enter the optional *interval* parameter, `syncstat` terminates after printing the one-line cumulative total of the line statistics.

If you enter a time interval (expressed in seconds), `syncstat` operates in an iterative sampling mode, sampling and then displaying line use data for the period specified. In this mode, `syncstat` does not output cumulative totals, but displays the incremental changes in the totals between iterations. For example, the command:

```
hostname# syncstat mcph1 10
```

would produce output similar to that shown below. Note that in this example, the display will be updated every ten seconds.

```
    ipkts   opkts  undrun  ovrrun   abort     crc   iutil   outil
       12      10       0       0       0       0      5%      4%
       22      60       0       0       0       0      3%     90%
       36      14       0       0       0       1     51%      2%
```

Note that issuing the `syncstat` command with a sampling time value adds two fields to the report: `iutil` and `outil`. These two fields report the level of use for the serial line, as a percentage of incoming bandwidth (`iutil`) and outgoing bandwidth (`outil`). These percentages may occasionally be reported as slightly greater than 100% because of inexact sampling times and differences in the accuracy between the system clock and the modem clock. If the percentage of use greatly exceeds 100%, or never exceeds 50%, then the speed value of the `syncinit` command probably does not equal the speed of the modem.

If you are experiencing communications problems, leave a `syncstat` command running on the console of the X.25 host machine so that you can see at a glance the recent history of loads and errors. For example, you might run `syncstat zss0 60` to get one-minute samples of activity on CPU port A.

If `syncstat` reports that line use is consistently near 100%, then you might need a faster line. An overloaded line can cause spurious HDLC and/or X.25 protocol timeouts. Also, watch for errors on the line, especially CRC errors in input packets. A small percentage of such errors can cause severe throughput reduction. These errors are almost always caused by troubles in the communication facilities.

If you see output packets but no input packets, then either the remote system is not initialized, or the line is not properly connected to the remote system.

If you see neither input nor output packets, then the physical layer was not successfully initialized.

If the `syncstat` reports:

```
syncstat: baud rate not set
```

then you have not initialized the line properly with the `syncinit` command.

9.5. Querying Network Interfaces with `ifconfig`

The `ifconfig` command is a standard SunOS command that is used to query and modify the flags associated with network interfaces, as well as to assign network addresses to network interfaces. See `ifconfig (8C)` in the *SunOS Reference Manual* for background on this command.

For the purposes of troubleshooting X.25 communication problems, `ifconfig` lets you check on the state of protocol devices that are part of the SunNet X.25 package. For example, to check on the hdlc device, enter:

```
hostname# ifconfig hdlc0
hdlc0: 0.0.0.0 flags=51<UP,POINTOPOINT,RUNNING>
```

If `ifconfig` displays the values UP, POINTOPOINT, and RUNNING, the link between the local host (DTE) and PSN (DCE) is properly configured; otherwise, `ifconfig` displays only the value POINTOPOINT.

For SunNet X.25, the devices you can check on are `hdlc`*n*, `std0` (for IP/dynamic and DDN X.25 Standard Service), `xpkt0`, and `xvc` (for static IP links).

With all of the above devices, `ifconfig` reports whether it is running or not. If a device is not running at a time when it should be, it might indicate that the device was not layered correctly with an `iflayer` command in the `/etc/sunlink/x25/rc` startup file. You can re-run `install.x25` to create a new startup file if you suspect this is true. Alternatively, you might not have set parameters—including the network address, if applicable—correctly for the level (packet, link, or physical) at which the non-running device resides.

9.6. The vcstat Command

The `vcstat` command allows you to monitor link and virtual circuit statistics, on a cumulative or periodic basis. The command has the following syntax

```
% /usr/sunlink/x25/vcstat [-L] [-n] [-l<interface>] [-i<interval>] [-p<period>]
```

The `vcstat` options are explained as follows:

`-L` Display link-related statistics instead of virtual circuit statistics. By default, `vcstat` displays virtual circuit statistics. Examples of displays for each type of statistics are shown below.

`-n` Display only cumulative (since reboot) statistics, instead of periodically updated display. By default, `vcstat` displays current statistics at 30-second intervals.

`-l`<*interface*>
Display statistics for the link specified by <*interface*>. This <*interface*> corresponds to the number specified for the `interface` parameter in your link configuration file.

`-i`<*interval*>
Sampling interval for display of cumulative statistics. If you omit this and the `-n` options, `vcstat` displays cumulative statistics at 30-second intervals.

`-p`<*period*>
Specifies the length of time, in minutes, `vcstat` will run when it is displaying current statistics. By default, `vcstat` displays statistics for 1440

minutes (24 hours).

Virtual Circuit Statistics

Without the `-L` option (that is, by default), `vcstat` displays statistics for all currently active virtual circuits, rather than for links. For example:

```
hostname% /usr/sunlink/x25/vcstat -i10

         Tue Sep 18 16:24:23 1990
   If LCN   Pgrp  State   Substate Sent  Recv  Remote address  MAC address
O   0 0x200     0 Data    0/0/0/0    130   130 129.144.133.2
O   0 0x201  1819 Data    0/0/0/0      3    10 10002244
I   1 0x202     0 Data    0/0/0/0      6     4 20009988        08:00:20:07:11:a1,0e
```

The fields and headings in the above example display are explained as follows:

Column 1
In the first column you see either `O`, for an outgoing call; `I`, for an incoming call; or `P`, for a permanent virtual circuit.

`If` Identifies the link (`xpkt` interface) over which the call was made. Corresponds to the value of the `interface` parameter in your link configuration file.

`Pgrp`
If there is a socket associated with a logical channel number, this field displays the number of the process group associated with that logical channel number.

`State`
Displays `Data` when the call is in the data transfer phase of the connection.

`Substate`
Displays four toggles (1 is true, 0 false). From left to right these are:

Sent Receive Not Ready
Sent Interrupt
Received Interrupt
Received Receive Not Ready

`Sent` and `Recv`
Displays the number of frames sent and received since the last reboot. (These counts are not reset if you stop and restart your link.)

`Remote Address`
This field displays the following types of addresses:

- IP addresses for virtual circuits used for IP (including DDN) connections.
- AEFs. This type of address is accompanied by a string, `osi`, `partial-osi`, or `non-osi`, indicating the type of AEF.
- X.121 addresses.

`MAC Address`
An 802.*x* SNPA address is present for virtual circuits running over LLC2.

Link Statistics

With the `-L` option, `vcstat` displays link-related, rather than virtual-circuit-related, statistics. It displays statistics for all currently active links. For example, to see link-related statistics, updated every 10 seconds, you enter:

```
hostname% /usr/sunlink/x25/vcstat -L -i10

         Wed Sep 19 08:56:52 1990
If Type State     SABM  Recv  Sent  Abort Crc   Over  Under MAC address
 0 hdlc UP        3     1112  1141  8     20    13    21
 1 hdlc UP        3     239   268   17    45    4     4
 2 llc  UP        1     601   589   4     12    0     0     08:00:20:07:55:fa,7e
```

The fields in the `vcstat` output are explained as follows:

`If` Identifies the link (`xpkt` interface) over which the call was made. Corresponds to the value of the `interface` parameter in your link configuration file.

`Type`
Can be `hdlc` or `llc`, identifying the type of connection. The designator `hdlc` indicates a serial-link connection, while `llc` indicates an LLC2 connection over a LAN.

`State`
Displays `UP` when the call is in the data transfer phase of the connection and `DOWN` when the call is being set up or taken down.

`SABM`
Indicates the number of Set Asynchronous Balanced Mode frames that have been sent. This type of frame is used to establish a frame-level connection.

`Recv` and `Sent`
Displays the number of frames sent and received since the last reboot. (These counts are not reset if you stop and restart your link.)

`Abort`
Displays the number of aborted received frames. Occurs when the local serial port received a sequence of eight consecutive ones, in violation of HDLC framing rules. Abort errors result from an interruption in the service provided by the link or from clocking problems. Such errors might also be caused by the software running over SunNet OSI. A small number of abort errors probably indicates a software problem rather than a broken link or a persistent clocking problem.

`Crc`
Reports the number of received frames with CRC (Cyclical Redundancy Check, an error detection method) errors. A CRC error is recorded when the checksum on a received frame is incorrect. CRC errors occur when there is a clocking problem (different rates on each side) or a noisy line.

`Over`
Reports the number of receiver overrun errors. Such errors occur when the local system is unable to accept data fast enough and the port hardware buffers overflow. A frame that is not completely received is aborted, triggering error recovery. Underrun errors can occur when the signaling rate in use on a link is too fast for the local system.

`Under`
Reports the number of transmitter underrun errors. Such errors occur when the local system is too busy to service the serial port hardware. A frame that is not completely sent is aborted, triggering error recovery. Underrun errors can occur when the signaling rate in use on a link is too fast for the local system.

`MAC Address`
An 802.*x* SNPA address is present only for virtual circuits running over LLC2. Note that there can be multiple links per `llc` device, one per destination. The LLC2 links time out if no virtual circuits use them.

9.7. The `x25trace` Command

The `/usr/sunlink/x25/x25trace` command allows you to obtain packet- and link-level traces on a per-call or per-interface basis. This command captures information about each packet and/or frame sent and received by SunNet X.25.

You must be `root` to run `x25trace`.

The output of `x25trace` is intended for the use of trained technical personnel, but even if you are not familiar with X.25 or HDLC communications, we encourage you to use the command when you are experiencing communications problems. The resulting output is valuable for people responsible for troubleshooting.

The `x25trace` command takes the form shown below:

```
# /usr/sunlink/x25/x25trace [options] [-i <interface>] <filter_expression>
```

The `x25trace` command must be run only in the foreground. If you want to capture its output in a file, use standard SunOS redirection to do so.

`x25trace` allows you to use specific MAC addresses as filters. For example, you can enter a command that has the effect of saying, “Trace all packets that travel over interface A between address 1 and address 2.” Given the length of MAC addresses, such commands can be extend beyond the width of a command line. Use the backslash (\) continuation character to continue `x25trace` commands beyond a single line.

`x25trace` remains in operation until you enter a Ctrl-C.

`x25trace` Options

The following options are available for all three types of tracing.

`-a` By default, `x25trace` displays the user data in hexadecimal for the highest protocol specified. This option tells `x25trace` to only display the number of bytes of user data and not display the data in hexadecimal. For example, if you enter:
`x25trace -a -i llc0 x25`,
`x25trace` displays only the number of bytes in X.25 packets sent and received over the `llc0` interface.

`-u` This option causes `x25trace` to buffer display output line-by-line, instead of the default operation of packet-by-packet.

-x%T{ This option causes `x25trace` to display entire packets in hexadecimal, in addition to its default operation of decoding the packet. This option is useful in troubleshooting malformed packets. With such packets, you often see an error message starting with two asterisks (`**`).

`x25trace` **Interfaces**

The following are the interfaces supported by `x25trace`:

`llc`*n*	the LLC interface for X.25 over LLC type 2.
`hdlc`*n*	the HDLC protocol used by X.25 over synchronous point-to-point lines
`snit_xpkt0`	the X.25 Packet Layer interface

`x25trace` **Expressions and Operators**

The following are the filter expressions you can use in an `x25trace` command line:

`betweenmac`	Trace only the packets or frames passing between the 802.*x* MAC addresses you specify.
`dstmac`	Trace only the packets or frames that have a destination address that is the 802.*x* MAC address you specify.
`llcf`	Trace only LLC frames using the SunOS `nit_pf (4M)` kernel filter. This filter reduces the load on your system and reduces the frequency of dropped frames.
`mac`	Trace packets and frames to/from the 802.*x* MAC address you specify. Use only when tracing on a LAN interface.
	multicast%T{ Trace only LLC2 packets that have multicast addresses.
`pdu_in`	Trace only incoming Protocol Data Units (PDUs).
`pdu_out`	Trace only outgoing Protocol Data Units (PDUs).
`srcmac`	Trace only the packets or frames that have a source address that is the 802.*x* MAC address you specify.

`x25lcn` *[+|<num>]* Used with a plus sign (+), this expression means trace only the packets that travel on the next logical channel set up. Used with a number, it means trace only the packets that travel on the logical channel identified by *<num>*.

You can use the logical operators `and`, `or`, and `not` with the expressions defined above.

`x25trace` Examples

This section presents examples of using `x25trace` for tracing incoming and outgoing packets. The first three commands are useful in tracing `x25manager` problems.

Tracing HDLC and X.25 on `hdlc`*n:*

The following command traces HDLC frames and X.25 packets as they are sent or received by HDLC.

```
hostname# x25trace -i hdlcn hdlc (or x25)
```

The following two commands are equivalent:

```
hostname# x25trace -i snit_xpkt0 x25
hostname# x25trace
```

Tracing LLC2 and X.25 on `llc`*n:*

```
hostname# x25trace -i llcn llc (or x25)
```

Note that the command above captures all LLC frames, including Unnumbered Information frames used to carry CLNP and ES-IS PDUs.

Tracing a single X.25 connection (logical channel number):

```
hostname# x25trace -i hdlcn x25lcn <lcn_num> x25
```

Tracing only the next X.25 connection being set up:

```
hostname# x25trace -i hdlcn x25lcn + x25
```

Tracing LLC2 frames on `llc`*n*:

```
hostname# x25trace -i llcn llc
```

Tracing X.25 packets to/from specific MAC addresses on `llc0`*:*

```
hostname# x25trace -i llc0 srcmac 8:20:0:1:2:3 x25
hostname# x25trace -i llc0 dstmac 8:20:0:1:2:3 x25
hostname# x25trace -i llc0 betweenmac 8:20:0:1:2:3 9:0:2b:18:21:5 x25
```

Trace incoming LLC2 frames using `nit_pf` *filter:*

```
hostname# x25trace -i llc0 llcf pdu_in
```

Trace outgoing LLC2 frames that are not multicast:

```
hostname# x25trace -i llc0 pdu_out not multicast
```

Redirecting trace of X.25 packets over an HDLC link to a file:

```
hostname# x25trace -a -i hdlc0 x25 > packets.record
```

9.8. X.25 Error Codes

A problem with an X.25 application might be a result of a problem communicating with the PSDN or a problem with the remote host. When you encounter a problem that causes your call to be cleared, you receive a diagnostic code in the form of two pairs of digits. The first pair indicates why the call was cleared; the second pair gives additional diagnostic information. The following tables specify meanings for these numeric codes. Table 9-4 specifies reasons for call-clearing. Table 9-5 specifies diagnostic information in addition to the call-clearing cause.

Use these tables as follows: when your connection is cleared unexpectedly and you receive a four-digit code, look up the first pair of digits in the first table; look up the second pair in the second table. A zero (`00`) call-clearing code indicates that the call-clearing request originated by the remote host. A non-zero call-clearing code indicates the call-clearing request originated by the PSDN.

For example, your call is cleared with a code `03 42`. As specified in Table 9-4, `03` indicates an invalid facility request. Table 9-5 tells you that `42` (hex) means that a facility parameter (that you used) is not allowed. This combination of messages tells you that:

1. The problem was detected by the PSDN, not by the remote host.
2. The parameter settings in the `x25params` file for the SunNet X.25 gateway are unacceptable to the PSDN.

Often, a diagnostic gives only a general clue as to the source of a problem. For example, your call might be cleared with the code `00 42`, indicating an invalid facility parameter setting. You might, in fact, have a facility parameter set (in your `x25mgr` configuration file) to a value outside of the range of what you subscribed to. However, you can also receive this message if the maximum I-frame size (the `max_frame_size` parameter in your `x25params` file) is set to a size lower than the PSDN expects. SunNet X.25 clears the call rather than allow it to start and possibly fail later.

If you receive error code `00 42` and are not subscribing to any facilities or are confident that your facilities parameters settings are correct, confirm that you and your PSDN agree on the maximum I-frame size. Remember that any adjustment of the maximum I-frame size (`max_frame_size`), an link-level parameter, might also require a change to the default packet size parameter (`def_packet_size`), at the packet level. Use `/usr/sunlink/x25/edit.x25params` to modify your link configuration file as necessary.

Table 9-4 *Causes for Call Clearing*

Hex Code	Decimal Code	Meaning
00	00	Originated by remote DTE
01	01	Number busy
09	09	Out of order
11	17	Remote procedure error
19	25	Reverse charging acceptance not subscribed
21	33	Incompatible destination
29	41	Fast select acceptance not subscribed
03	03	Invalid facility request
0B	11	Access barred
13	19	Local procedure error
05	05	Network congestion
0D	13	Not obtainable
15	21	RPOA out of order

In Table 9-4, note the four categories of codes, separated by the double horizontal lines:

- The first category, code 00, indicates that the call-clearing request originated with the remote host. The remaining codes, in all other categories, indicate that the call-clearing request originated within the PSDN.
- Codes in the second category indicate problems between the PSDN and the remote host.
- Codes in the third category indicate problems between the local host and the PSDN.
- Codes in the fourth category indicate problems within the PSDN.

Table 9-5 *X.25 Diagnostic Codes*

Hex Code	Decimal Code	Meaning
00	**00**	**No additional information**
01	01	Invalid P(S) (send-packet sequence number)
02	02	Invalid P(R) (receive-packet sequence number)
10-1F	**16-31**	**Invalid packet type**
20	**32**	**Packet not allowed**
21	33	Unidentifiable packet
22	34	Call on one-way logical channel
23	35	Invalid packet type on a permanent virtual circuit
24	36	Packet on an unassigned logical channel
25	37	*Reject* not subscribed to
26	38	Packet too short
27	39	Packet too long
28	40	Invalid general format identifier
29	41	Restart with nonzero in bits 1-4, 9-16
2A	42	Packet type not compatible with facility
2B	43	Unauthorized interrupt confirmation
2C	44	Unauthorized interrupt
30	**48**	**Timer expired**
31	49	For incoming call
32	50	For clear indication
33	51	For reset indication
34	52	For restart indication
40	**64**	**Call setup problem**
41	65	Facility code not allowed
42	66	Facility parameter not allowed
43	67	Invalid called address
44	68	Invalid calling address

Note in the table above that hexadecimal codes 0, 20, 30, and 40 (decimal 0, 32, 48, and 64) stand for general messages that include the specific conditions indicated by the codes that follow them. Your PSDN might return only the codes that stand for the general messages.

Table 9-6 lists X.25 diagnostic codes that you might receive when you are running SunNet X.25 over the Connection-Oriented Network Service (CONS).

Table 9-6 *CONS Diagnostic Codes*

Hex Code	Decimal Code	Meaning
Sent by CONS		
E0	224	Unspecified
E1	225	Disconnect—transient
E2	226	Disconnect—permanent
E3	227	Reject—transient
E4	228	Reject—permanent
E5	229	QOS not available—transient
E6	230	QOS not available—permanent
E7	231	NSAP unreachable—transient
E8	232	NSAP unreachable—permanent
E9	233	Unspecified Reset
EA	234	Reset due to congestion
EB	235	Unknown NSAP
Sent by CONS user (TP0/TP2)		
F0	240	Unspecified
F1	241	Normal disconnect
F2	242	Abnormal disconnect
F4	244	Reject—transient
F5	245	Reject—permanent
F6	246	No QOS—transient
F7	247	No QOS—permanent
F8	248	Incompatible NS user data
FA	250	User Reset

Note that you receive the reset codes, hexadecimal numbers E9, EA, and FA, in Reset Request/Indication packets. You receive the remaining codes in Clear Request/Indication packets.

For both its X.29 and PAD applications, SunNet X.25 creates a file, `/var/adm/x29debug,` to which it sends error messages. These messages are intended for the use of Sun technical personnel. If you experience problems while using the X.29 or PAD applications, check this file for messages. If messages are present, forward them to your local Sun customer service representative.

9.9. IP Routing Problems

The following steps might be required to diagnose problems that involve the routing between your local network and the PSDN.

Check Whether Route Exists to and from Remote Host

Use the `ping (8C)` command to determine whether a route exists from your machine to a remote machine (`hermes` in the example below) and from the remote machine back to your machine. For example:

```
hostname# ping hermes
```

If the result is:

```
hermes is alive
```

then packets can be sent to and received from `hermes`. If not, a routing problem exists at some point between the local and remote hosts.

For more detailed information, use the `-s` option. For example:

```
hostname% ping -s hermes
PING hermes: 56 data bytes
64 bytes from hermes (129.144.133.2): icmp_seq=0. time=239. ms
64 bytes from hermes (129.144.133.2): icmp_seq=1. time=160. ms
64 bytes from hermes (129.144.133.2): icmp_seq=2. time=160. ms
64 bytes from hermes (129.144.133.2): icmp_seq=3. time=160. ms
64 bytes from hermes (129.144.133.2): icmp_seq=4. time=160. ms
<Ctrl-C>
----hermes PING Statistics----
6 packets transmitted, 5 packets received, 16% packet loss
round-trip (ms)  min/avg/max = 160/175/239
```

Check Local IP Routing Table

Use the `netstat (8C)` command as follows:

```
hostname# netstat –r
```

to display the local IP routing tables. Make sure there is a routing table entry for each possible destination network. For example,

```
hostname# netstat -r
```

would produce output similar to the following:

```
Destination    Gateway         Flags    Refcnt Use        Interface
default        LBL-MILNET-GW.A U        256    595818     std0
b14-eng-net    gogo            U        203    568090     le0
loopback       localhost       UH       3      82411      lo0
```

where:

`Destination`	names the destination host or network.
`Gateway`	names the system to be used in forwarding packets.
`Flags`	The flag `U` indicates that the state is "up," `G` indicates that the route is to a gateway, and `H` indicates that the route is to a host.
`Refcnt`	gives the current number of active users of the route.
`Use`	gives a count of the number of packets sent on the route.
`Interface`	names the route's network interface.

This information makes it possible to verify whether a routing table entry exists for a particular destination network.

If you want to find out what the next hop (gateway) to a destination is, use a command such as:

```
hostname# netstat -r -n | grep <dest_net_number>
```

The `-n` option makes `netstat` display the `Destination` and `Gateway` fields as Internet addresses, rather than names. *dest_net_number* is the network number of the host that is your ultimate destination.

Routing over IP/Dynamic or DDN/Dynamic Interfaces

Seemingly random IP routing failures can occur while using the IP/dynamic or DDN/dynamic interface (`std0`) if the IP address of this interface has the same network number as the primary interface (usually the Ethernet interface) of your machine. In such a situation, you might receive a message such as:

```
syslog: deleting route to interface std0 (timed out)
```

Use the `netstat -i` command to check the addresses of the `std0` and primary interfaces. For example:

```
hostname% netstat -i
Name  Mtu  Net/Dest      Address        Ipkts   Ierrs Opkts  Oerrs Collis Queue

le0   1500 b14-spd-b     ariditas       1360097 102   222972  2    20142  0
std0  1004 b12-alm2      ariditas       229     0     343     0    0      0
lo0   1536 loopback      localhost      48551   0     48551   0    0      0
```

In the output above note that `le0` and `std0` have the same address. (The hostname `ariditas` is equivalent to a numeric IP address.) To correct such a problem, you must obtain an IP address with a locally unique network number for the `std0` interface. Then, as `root`, you must edit the file `/etc/sunlink/x25/x25manager.configuration`. In this file, find a line such as the following:

```
shell /usr/etc/ifconfig std0 ariditas netmask + up
```

Change the IP address (`ariditas` in the example above) to the new IP address. Then enter a sequence of commands such as the one listed below. The process id number, `6174` below, is only an example.

```
hostname% ps ax | grep x25mgr | grep -v grep
 6174 ?  S     0:00 x25mgr x25manager.configuration
hostname% su
hostname# kill -9 6174
hostname# cd /etc/sunlink/x25
hostname# x25mgr x25manager.configuration
hostname# exit
hostname%
```

Re-run `netstat -i` to confirm that the changed address is now in use.

9.10. Checking the HDLC Service

If you are using only the HDLC interface, you can also use `hdlcstate` to check the physical and link layers of the `hdlc` device. `hdlcstate` accepts the name of an `ifd` device as its only argument, as in the following command:

```
hostname% /usr/sunlink/x25/hdlcstate /dev/ifd0
Link state: UP
Statistics:
        SABMs sent 4
        Packets: output 15144, input 15272
        Total bytes: sent 4669337, received 4994754
        Errors: abort 1, crc 0, overrun 0, underrun 0
Parameters:
        port type "dte", window size 7, max frame size 134 bytes
        T1=10000ms(retransmit timer), N2=10(max retransmits)
        T2=0ms(acknowledge delay), T3=20000ms(max line idle time)
```

The values displayed are cumulative, that is, they are only reset at boot time. `Packets`, `Total bytes`, and `Errors` are all values that come from the physical level. Along with statistics, `hdlcstate` displays the current HDLC configuration parameters.

9.11. The `ifsetmtu` Command

The SunNet OSI and SunNet X.25 products both include a utility, `ifsetmtu`, that you use to set Maximum Transmission Unit (MTU) sizes. This utility is stored in the `/usr/sunlink/sync` directory following product installation. `ifsetmtu` has the following syntax:

```
# /usr/sunlink/sync/ifsetmtu <interface> <mtu_size>
```

The <*interface*> is a network interface, such as `xncl0`, `std0`, or an `xvc`*n* interface. The <*mtu_size*> is the MTU size in bytes.

You must be `root` to run `ifsetmtu`.

Use the `netstat -i -a` command to view all of your network interfaces with their MTU sizes.

As an example, we change the MTU size for the `xncl0` interface from its default of 512 bytes to 2048 bytes.

```
hostname% netstat -i -a | grep xncl | awk '{print $1 " " $2}'
xncl0* 512

become super-user

hostname# /usr/sunlink/sync/ifsetmtu xncl0 2048
hostname# netstat -i -a | grep xncl | awk '{print $1 " " $2}'
xncl0* 2048
```

Note that the changed MTU size takes effect immediately.

A

PSDN-specific Parameter Settings

This appendix lists common settings for link configuration file parameters that are specific to major PSDNs in Europe, Asia, and North America. Figures presented here offer guidelines. In particular, logical channel number (LCN) ranges can vary for your connection to the PSDN. You should confirm the numbers presented here with your PSDN administration.

Table A-1 lists all of the PSDNs we cover, with the network's location and the name of the organization responsible for it. For each network, Table A-2 presents useful information for link configuration files. sp 2p

Table A-1 *PSDNs: Countries and Organizations*

Network Name	*Country*	*Responsible Organization*
Accunet	United States	AT&T Corp.
Datapac	Canada	National PTT
Datapak-2	Sweden	National PTT
Datex-P	Germany	National PTT
DCS	Belgium	National PTT
DDX	Japan	National PTT
Eripax	Sweden	Telefon AB LM Ericsson
IBERPAC	Spain	National PTT
JANET (Joint Academic NET-work)	United Kingdom	consortium of private research and education organizations in the UK
LUXPAC	Luxembourg	National PTT
PSS (Packet Switched Stream)	United Kingdom	British Telecom
Telenet	United States	SPRINT/Telenet Corp.
Transpac	France	National PTT
Tymnet	United States	Tymnet Corp.

Table A-2 *Configuration File Information*

Network Name	*2-way LCN Range*	*DTE Address Breakdown*	*Subaddress Digits Allowed*	`calling_address` *Setting*	`called_address` *Setting*	`address_prefix` *Setting*
Accunet		4-digit DNIC, 8-digit NTN	0-2	no_dnic	no_dnic	0
Datapac						
Datapak-2	1-8	4-digit DNIC, 7-digit NTN	0-3	no_dnic	no_dnic	0
Datex-P	1-255	4-digit DNIC, NTN up to 10 digits or 11, if DCC is used	0-3	no_dnic	no_dnic	0
DCS	1-16	4-digit DNIC, 7-digit NTN	2	full	full	not used
DDX-Plan 1†	1-15	4-digit DNIC, 7-digit NTN	0	full	full	not used
DDX-Plan 2†	0-32	4-digit DNIC, 8-digit NTN	0-2	full	full	not used
Eripax						
IBERPAC	1-120	4-digit DNIC, 9-digit NTN	0-2	no_dnic	no_dnic	0
JANET	0x400-0x40f	4-digit DNIC, 8-digit NTN	0-2	full	full	not used
LUXPAC	1-16	4-digit DNIC, 7-digit NTN	2	subaddress	full	not used
PSS	0x400-0x40f	4-digit DNIC, 8-digit NTN	0-2	full	full	not used
Telenet	1-32	4-digit DNIC, 3-digit Area Code, 2-digit filler (00), 3-digit NTN	2-4	full	full	not used
Transpac	start with 0	4-digit DNIC, 8-digit NTN	0-3	subaddress	no_dnic	not used
Tymnet	1-30	4-digit DNIC, 4-digit NTN	2	full	full	not used

† DDX Plan 1 allows connections to the Venus-P international PSDN via the DDX-P network. DDX Plan 2 allows for direct connections to the Venus-P network.

B

The Link Configuration File

This appendix describes each parameter in the link configuration file. The link configuration file is a text file that has three sections, interface, frame (HDLC or LLC2), and packet, corresponding to the three layers—physical, frame (HDLC or LLC2), and packet—that make up X.25.

B.1. Introduction

The `x25config` program, stored in `/usr/sunlink/x25`, uses the parameters in your link configuration file to set up a link. (See Section B.5 for more details on `x25config`.) The installation script `/usr/sunlink/x25/install.x25` creates a link configuration file for each of your links, using your responses to the script's queries, and stores the file(s) in `/etc/sunlink/x25`. Following installation, use the script `edit.x25params`, stored in `/usr/sunlink/x25`, to make changes to your link configuration file(s).

You must have a link configuration file for each X.25 link you use. Links are numbered in a range from 0 to one fewer than the number of links you have. For example, if you have a single link, it is designated link 0. If you have four links, they are numbered 0 through 3.

`install.x25` names your link configuration files `x25params`*n*, where *n* is a link number, so that `x25params0` refers to link 0, `x25params1` refers to link 1, and so forth. For LLC2 links, the names are of the form `x25params`*n*`_llc`.

When you use `install.x25` or `edit.x25params`, take care that your parameter values *exactly* match those supplied by the PSDN providing the subscription service. See Appendix A for PSDN-specific configuration parameters for some major PSDNs.

Note that the link configuration file is important only to those who use the `x25config` program that is supplied with SunNet X.25. If you use a client-supplied application program instead of `x25config`, you do not use the link configuration file described here.

B.2. Mandatory and Optional Parameters

SunNet X.25 supplies default values for most of the parameters that can appear in a link configuration file. Any optional parameters that have default values that are appropriate for your link need not be present in your link configuration file. (The installation script `install.x25` comments out optional parameters whose values are not modified when you run the script.) For example, for an HDLC link for which all default values are appropriate, the link configuration file contains only:

```
interface <link_num>
link hdlc
x25
address <x121_addr>
lcn_two_way 512 575
```

The analogous file for an LLC2 link contains:

```
interface <link_num>
link llc
x25
address null
lcn_two_way 1 4095
```

The parameters listed in the above examples, as well as all optional parameters, are described in Section B.4.

To display all of the values for a link, including those specified in the link configuration file and those supplied by SunNet X.25 by default, use the `-d` option to `x25config`, specifying your link configuration file in the command line. For example:

```
hostname# /usr/sunlink/x25/x25config -d x25params0
```

See Section B.5 for a description of `x25config`.

B.3. Notes on Local X.121 Host Address

For HDLC and some LLC2 configurations, your host address is the X.121 address of your machine. You specify this address in the `address` parameter in the packet layer section of you link configuration file (see Section B.4.3). An X.121 address consists of a DNIC (Data Network Identification Code); NTN (Network Terminal Number); and, optionally, a subaddress. The DNIC is usually four binary-coded-decimal (BCD) digits; a host address can be up to 14 BCD digits. If you do not have an X.121 address (common on LLC2 links), `address` is set to `null`.

As entered in your link configuration file, the DNIC, NTN, and subaddress are separated by hyphens, for example, `4042-3831-00`.

The `address` parameter does not require a subaddress component. If a subaddress is present, SunNet X.25 will use it as the subaddress in the source (calling) address field of outgoing Call Request packets, unless the application making the call specifies its own subaddress. If `address` does not have a subaddress, SunNet X.25 will use a zero-length subaddress for the source address field of outgoing Call Request packets, unless the application initiating the call specifies otherwise.

PSDNs vary in their requirements for how much of the source (calling) and destination (called) addresses go into a Call Request packet. Some require that the DNIC always be sent, while others require that the DNIC be sent only for long-distance calls. Certain PSDNs require that a zero or other digit be prepended to a full X.121 address.

To handle the different PSDNs' requirements, SunNet X.25 provides the `called_address__opt`, `calling_address_opt`, and `address_prefix` parameters that `install.x25` sets in your link configuration file. These parameters determine how much of the calling and called addresses are used in the Call Request packet and whether a digit is prepended to the addresses.

To establish successful X.25 communication, you must specify the full host address (or `null` for LLC2 links) for the `address` parameter and set `called_address__opt`, `calling_address_opt`, and `address_prefix` according to the requirements of your PSDN.

B.4. Configuration Parameters

The following subsections describe the configuration parameters for the three sections—interface, link, and packet-layer—of a link configuration file. The order of parameters within a section is not significant.

Interface Section

The first section of the link configuration file contains the parameter that identifies your link.

`interface` *<id>*

> The integer *<id>* corresponds to the number of an `xpkt` device. SunNet X.25 uses one `xpkt` device for each HDLC and LLC2 link you have. These devices are numbered starting with zero. Your highest `interface` number is one less than your total number of links.

Link Section

The second section of the link configuration file contains the parameter settings for the frame level, either HDLC (LAPB) or LLC2.

The start of this section is indicated by the `link` parameter, which is the only mandatory parameter in the section, for both HDLC and LLC2 links. You need other parameters only if the default settings for your link type are not appropriate for your link.

The `link` parameter has the following form:

`link` *<link_type>* *<control>*

> where *<link_type>* is `llc` or `hdlc` and *<control>* is `normal` or `extended`. Only the *<link_type>* argument is mandatory. The *<control>* argument is optional for HDLC links. The default for HDLC links is `normal`. For LLC2 links, the *<control>* argument is always `extended`.
>
> When *<control>* is `normal`, the link level operates in normal mode. In this mode, the default value of 7 for the `window_size` parameter (see Table B-1) for HDLC links is appropriate. When *<control>* is `extended`, the link level operates in extended mode. For HDLC links, you should set the `window_size` parameter to 127.

For an HDLC link that uses extended mode, the `link` parameter is set as follows:

```
link hdlc extended
```

The remaining link section parameters are described in Table B-1.

Table B-1 *Link Section Parameters*

Name	*Argument*	*HDLC Default*	*LLC2 Default*	*Description*
`lsap`	<*lsel*>	NA	0x7e	An OSI Link Selector (LSEL) used only with LLC2 links. The LSEL must be less than 254 (decimal) and must be an even number.
`speed`	<*bits_per_second*>	9600 bps	NA	The signaling rate used on the of the synchronous serial line. Note that `speed` should correspond to the line speed specified in the `syncinit` command for this particular link. (In the startup file `/etc/sunlink/x25/rc`, `install.x25` sets up a `syncinit` command appropriate for your link.) The actual line speed is provided by the modem or, if you are connecting two machines with a null modem cable, by one or both machines. This parameter is used strictly for internal calculations.
`tick`	<*milliseconds*>	500	500	The resolution of the clock that controls the timing of the timer parameters, such as `poll_timeout`, `ack_delay`, and `idle_timeout`.
`poll_timeout`	<*milliseconds*>	4500	2000	Corresponds to retransmission timer, `T1`. This parameter is the length of time, in milliseconds, that your host waits for a response to a command (poll) before initiating error recovery. Unacknowledged command frames are retransmitted whenever this timer expires.
`idle_timeout`	<*milliseconds*>	18000	10000	Corresponds to the idle link timer, `T3`. This is the period of idle time after which a keep-alive poll is sent.
`busy_timeout`	<*milliseconds*>	6000	6000	The length of time that your host will wait for the clearance of a busy condition in the remote host.
`rej_timeout`	<*milliseconds*>	6000	6000	The length of time that your host will wait for a Reject condition to be cleared by the remote side.
`ack_timeout`	<*milliseconds*>	6000	6000	The acknowlege-command timer. This is the length of time your host will wait for an acknowledgement to one or more outstanding data frames before resetting the link with a SABM frame.
`ack_delay`	<*milliseconds*>	2000	500	Corresponds to the acknowledge delay timer, `T2`. It specifies the maximum delay before the receiver of a frame transmits an acknowledgement frame.
`max_frame_size`	<*number_bytes*>	134	1030	Specifies the maximum I-frame size in bytes. This parameter can be set to any value but it must exceed the value of the `def_packet_size` parameter (see following subsection) by at least five (normal HDLC mode) to eight (extended HDLC mode) bytes.
`max_retries`	<*number_retries*>	10	10	Corresponds to the `N2` parameter, which specifies the number of times a frame should be retransmitted after the `T1` timer (`poll_timeout`) has expired.

Table B-1 *Link Section Parameters— Continued*

Name	*Argument*	*HDLC Default*	*LLC2 Default*	*Description*
`window_size`	<*number_windows*>	7	127	The window size parameter, `k`, that specifies the maximum number of I-frames that can be unacknowledged at any time. For HDLC links, if you specify extended mode for the `link` parameter, you should set `window_size` to 127. (LLC2 links always operate in extended mode.)

Packet-level Section

The third section of the link configuration file contains the parameter settings for the packet level. The start of this section is indicated by the mandatory `x25` parameter. The only other mandatory parameters in the section are `address` and one of the logical channel number parameters, `lcn_incoming`, `lcn_two_way`, or `lcn_outgoing`.

The `x25` parameter has the following form:

`x25` <*version*> <*format*> <*mode*>

where <*version*> can be `1980` or `1984`. The default setting for HDLC links is `1984`. LLC2 links always use the `1984` setting. Use the `1980` setting if your PSDN supports only the functions of the 1980 X.25 recommendation.

The value for <*format*> is either `normal` (the default) or `extended`. With `normal`, packet sequence numbers range from 0 to 7. With `extended`, packet sequence numbers range from 0 to 127.

The value for <*mode*> can be `dte`, the default for HDLC links; `dynamic`, the default for LLC2 links, and `dce`. For LLC2 links, the mode is always determined dynamically, regardless of your setting for <*mode*>.

For HDLC links, if you are connecting to a PSDN, set <*mode*> to `dte`. If you are connecting a Sun to another machine (Sun or non-Sun) using a null modem cable or modem eliminator, set <*mode*> so that one machine is different from the other, either `dte` or `dce`.

If you set the <*mode*> argument to accommodate a null modem connection, take care to not omit the <*version*> and <*format*> arguments. While all of the arguments to `x25` are optional, if you include the <*mode*> argument, it must be preceded by <*version*> and <*format*>.

The `address` parameter has the following form:

`address` <*address*>

where <*address*> is the local X.121 address for HDLC links or `null` for LLC2 links. See Section B.3 for a discussion of this parameter.

The link configuration file has the following parameters that determine how calling and called addresses appear in Call Request packets:

`called_address_opt` *[*`full` | `no_dnic`*]*

Used only for calls within the PSDN to which the link is directly connected. Used to determine how the destination (called) address field in a Call Request packet is set. If `full`, the complete (DNIC, NTN, subaddress)

X.121 address is used. If `no_dnic`, the DNIC portion is dropped for calls within the local PSDN. The default is `full`.

For this parameter to work correctly, you must specify the full destination address when originating a call. For calls outside the local PSDN, SunNet X.25 always uses the full X.121 address specified for the destination.

`calling_address_opt` *[*`full` | `no_dnic` | `subaddress`*]*
: Used for both local and long-distance calls. Determines how the source (calling) address field in a Call Request packet is set. If `full`, the complete (DNIC, NTN, subaddress) X.121 address is used. If `no_dnic`, the DNIC portion is dropped. If `subaddress`, only the subaddress (if present) is used. The default is `full`.

For this parameter to work correctly, you must specify the full X.121 source address in the `address` parameter.

`address_prefix` <*prefix*>
: If this parameter is present, <*prefix*> is prepended to calling and called addresses in Call Request packets whenever those addresses have a DNIC present.

You must have one of the logical channel number (lcn) parameters described below present in your link configuration file. The absence of an lcn range means that the range is null. These parameters have the following forms:

`lcn_incoming` <*number*>-<*number*>

`lcn_two_way` <*number*>-<*number*>

`lcn_outgoing` <*number*>-<*number*>
: In `install.x25` you enter `empty` for an empty range. The valid range for lcn's is 1-4095.

Where you have more than one type of logical channel, the distribution of lcn's should be such that the range of one-way outgoing lcn's should be higher than the range of two-way lcn's, which, in turn, should be higher than the range for one-way incoming lcn's. Specifically:

- The highest one-way outgoing lcn should be greater than or equal to the lowest one-way outgoing lcn.
- The lowest one-way outgoing lcn should be greater than or equal to the highest two-way lcn.
- The highest two-way lcn should be greater than or equal to the lowest two-way lcn.
- The lowest two-way lcn should be greater than or equal to the highest one-way incoming lcn.
- The highest one-way incoming lcn should be greater than or equal to the lowest one-way incoming lcn.

SunNet X.25 assumes lcn's lower than your lowest, non-zero lcn range can be assigned to Permanent Virtual Circuits (PVCs).

The remaining link section parameters are described in Table B-2.

Table B-2 *Packet-layer Section Parameters*

Name	*Argument*	*HDLC Default*	*LLC2 Default*	*Description*
`def_window_size`	<*number_packets*>	2	7	The default window size for every virtual circuit. The value of this parameter specifies the number of unacknowledged I-frames that can be outstanding at any given time. You can specify a window size from 1-7 when the `x25` parameter is set to `normal` and from 1-127 when `x25` is set to `extended`.
`def_throughput`	<*index_num*>	9600 (10)	9600 (10)	The default maximum throughput, as specified in the throughput facility, for every virtual circuit. This value does not necessarily correspond to the signaling rate used on the line. When a Sun is acting as a DTE or DCE, the value of `def_throughput` has no effect on the local machine and is irrelevant. However, the value might be of importance to the DCE to which the DTE is connected. The DCE might use the value of this parameter to control the flow of data through the virtual circuit. When a Sun, acting as a DTE, receives an Incoming Call, it does not negotiate a throughput greater than the value specified for `def_throughput`. The default value for `def_throughput` is 10 (9600 bps). Other possible values are enumerated in Table B-3, immediately following this table. Note that not all PSDNs support the throughput facility.
`def_packet_size`	<*number_bytes*>	128	1024	The default packet size in bytes. In addition to the defaults, acceptable values for this parameter are 16, 32, 64, 256, and 512, 2048, 4096. If you change this parameter, adjust the `max_frame_size` parameter in the link section so that it remains at least 5-8 bytes larger than `def_packet_size`.
`restart_timeout`	<*number_seconds*>	180	180	Corresponds to the Packet Layer `T20` parameter. This is the timeout period to wait after sending a Restart Request packet. If SunNet X.25 receives no response to a restart request by the time this timer expires, it sends the Restart Request packet again. Values from 1-255 seconds are supported for this parameter.
`call_timeout`	<*number_seconds*>	200	200	Corresponds to the Packet Layer Call Request timeout parameter, `T21`. If SunNet X.25 receives no response to a Call Request packet in `call_timeout` seconds, SunNet X.25 transmits a Clear Request packet. Values from 1-255 seconds are supported for this parameter.
`reset_timeout`	<*number_seconds*>	180	180	Corresponds to the Packet Layer Reset Request timeout parameter, `T22`. If SunNet X.25 receives no response to a Reset Request packet in `reset_timeout` seconds, it transmits a Clear Request, for an SVC, or a Reset Request, for a PVC. Values from 1-255 seconds are supported for this parameter.
`clear_timeout`	<*number_seconds*>	180	180	Corresponds to the Packet Layer Clear Request timeout, `T23`. If SunNet X.25 receives no response to a Clear Request packet in `clear_timeout` seconds, it sends another Clear Request packet. Values from 1-255 seconds are supported for this parameter.
`restart_retries`	<*number_retries*>	3	3	This parameter specifies the number of times SunNet X.25 should transmit a Restart Request packet once the `restart_timeout` timer has expired. Values from 1-255 are supported for this parameter.

Table B-2 *Packet-layer Section Parameters— Continued*

Name	*Argument*	*HDLC Default*	*LLC2 Default*	*Description*
`call_retries`	<*number_retries*>	1	1	This parameter specifies the number of times SunNet X.25 should transmit a Call Request packet once the `call_timeout` timer has expired. Values from 1-255 are supported for this parameter. However, this value should be set to 1 to conform with CCITT specifications.
`reset_retries`	<*number_retries*>	3	3	This parameter specifies the number of times SunNet X.25 should transmit a Reset Request packet once the `reset_timeout` timer has expired. Values from 1-255 are supported for this parameter.
`clear_retries`	<*number_retries*>	3	3	This parameter specifies the number of times SunNet X.25 should transmit a Clear Request packet once the `clear_timeout` timer has expired. Values from 1-255 are supported for this parameter.
`send_buffer_size`	<*number_bytes*>	2048	2048	This parameter specifies the number of bytes of buffer space available to a SunNet X.25 application that is sending data. If this amount is exceeded, the application is blocked until sufficient space is available for more data.
`recv_buffer_size`	<*number_bytes*>	2048	2048	This parameter specifies the number of bytes of buffer space available to receive incoming X.25 packets. Once this data space has been filled, SunNet X.25 transmits Receive Not Ready (RNR) to flow control incoming data on the virtual circuit. SunNet X.25 transmits Receive Ready (RR) when more data can be accepted for the virtual circuit.
`D_bit`	`yes` *or* `no`	no	no	This parameter specifies whether the D bit is used in calls that originate from SunNet X.25. If you set this parameter to 1, the D bit is used; if 0, the D bit is not used. For incoming calls, SunNet X.25 always accepts packets with the D bit turned on.

Table B-3 lists the range of allowable values for the `def_throughput` parameter, which is described in Table B-2.

Table B-3 *Throughput Facility Values*

`3 :`	75 bps	`8 :`	2400 bps
`4 :`	150 bps	`9 :`	4800 bps
`5 :`	300 bps	`10 :`	9600 bps
`6 :`	600 bps	`11 :`	19200 bps
`7 :`	1200 bps	`12 :`	48000 bps

B.5. The `x25config` Command

The following information is primarily for reference purposes. The SunNet X.25 installation script inserts an `x25config` command appropriate for your configuration in the X.25 startup file, `/etc/sunlink/x25/rc`.

Command Syntax

`x25config` begins the operation of SunNet X.25. It has the following general form:

```
hostname# /usr/sunlink/x25/x25config [-on|-off] [-adp] [-llinkid] config_file
```

The command parameters are described as follows:

`-on|-off`
: Specifies that the link be activated (`-on`) or deactivated (`-off`).

`-a` Specifies that `x25config` exit without waiting to confirm that X.25 software has established packet service to the network. (See further explanation of this very useful option, below.)

`-d` Display all of the values for a link, including those specified in the link configuration file and those supplied by SunNet X.25 by default. Also, performs the syntax and range checking you receive with the `-p` option.

`-l`<*linkid*>
: Designates a link number. You specify this parameter only in conjunction with the `-off` parameter. If you specify the link number with `-off`, you can omit the configuration file name.

`-p` Parse mode. Specified in conjunction with the name of a link configuration file. `x25config` checks the specified file and either returns silently if file syntax and parameter ranges are correct or returns with an error message if it finds an error.

As an example, assume you want to start link 2, using the `-a` option. The configuration file associated with this link is `/etc/sunlink/x25/x25params2`. You enter the following command:

```
hostname# x25config -a /etc/sunlink/x25/x25params2
```

Bringing Down a Link

Use an `x25config` command like the following to bring down a link, from the packet layer down:

```
hostname# x25config -off -l<link_num>
```

Alternatively, you can specify the link configuration file for the link you want to bring down, instead of the link number.

The `-a` Option

When you invoke `x25config` without the `-a` option, it waits until the link level makes a connection and the X.25 level completes a restart procedure before returning. If you specify the `-a` option, `x25config` returns immediately, while the attempt to make a connection proceeds in the background. An example of the benefit of using `-a` occurs when a remote host is down. If you invoked x25config without -a, x25config waits until the link level times out before returning. This might be several minutes. With -a, x25config returns immediately, allowing you to enter another command.

An alternative to using the -a option is to run x25config in the background. You can run multiple instances of x25config at the same time.

C

Link Configuration File for DDN X.25 Standard Service

This appendix is a reference for those SunNet X.25 users who are using X.25 Standard Service to connect to the DDN. The link configuration file is a text file containing three sections, interface, link, and packet, that correspond to the three layers—physical, HDLC, and packet—that comprise X.25. If you need to make any changes to it following installation, use the script `/usr/sunlink/x25/edit.x25params`.

The link configuration file is input to the `x25config` program, which does the work of configuring your link. The `x25config` program is described in Section B.5.

C.1. Guidelines for Link Configuration File Settings

Check with the administrator responsible for the PSN before changing the settings of any parameters.

In the link configuration file, you can specify integers in decimal, octal, or hexadecimal. You must use C programming language conventions for expressing integers. For example, the decimal number 25 in hexadecimal is 0x19 and in octal is 031. Note that 08 is an invalid octal number; it should be 010.

C.2. Mandatory and Optional Parameters

SunNet X.25 supplies default values for most of the parameters that can appear in a link configuration file. Any optional parameters that have default values that are appropriate for your link are not present in your link configuration file. For example, for a DDN connection for which all default values are appropriate, the link configuration file contains only:

```
interface 0
link hdlc
x25
address <x121_addr>
lcn_two_way 512 575
```

The parameters listed in the above example, as well as all optional parameters, are described in Section B.4.

To display all of the values for a link, including those specified in the link configuration file and those supplied by SunNet X.25 by default, use the `-d` option to `x25config`, specifying your link configuration file in the command line. For example, if you are in `/etc/sunlink/x25`, enter:

```
hostname% /usr/sunlink/x25/x25config -d x25params0
```

Section B.5 explains the use of `x25config` in detail.

C.3. Matching Parameter Settings with the PSN

Settings for certain parameters in the configuration file must match the settings of the PSN to which you are connecting. Obtain these settings from the PSN's administrator. The parameters that must match are listed below. For X.25 Standard Service, you can leave the parameters not listed below with their default values.

Table C-1 *Parameters Requiring PSN-Host Match*

Parameter	Description
	Link Parameters
`speed`	Signaling rate in bits per second. Used for diagnostic purposes.
`max_frame_size`	Maximum I-frame size in bytes. Set to 1032.
`window_size`	Window size. Default value of 7 is maximum for X.25 Standard Service. Optionally, can be number in the range 1 to 6.
	Packet Parameters
`address`	X.121 address of the SunNet X.25 machine. For X.25 Standard Service, derived from Class A or Class B IP address as assigned by DDN.
`x25` *<version><format><mode>*	Set to `x25 1980 normal dte`.
`def_window_size`	Window size for all virtual circuits. Maximum that can be negotiated for X.25 Standard Service is 7. (Default is 2.)
`def_packet_size`	Packet size in bytes. For X.25 Standard Service, maximum packet sizes of 16, 32, 64, 128, 256, 512, and 1024 bytes can be negotiated. (Default is 128.)
`lcn_incoming`	Range of one-way incoming logical channel numbers.
`lcn_two_way`	Range of two-way logical channel numbers.
`lcn_outgoing`	Range of one-way outgoing logical channel numbers.
`calling_address_opt`	Set to `full`.
`called_address_opt`	Set to `full`.
`address_prefix`	Not used.

See Appendix B for an description of each of the link configuration parameters.

C.4. Deriving DDN X.25 Addresses for X.25 Standard Service

This section tells you how to derive an X.25 (X.121) address from a Class A Internet address so that you can fill in the `host_address` parameter in your link configuration file.

The DDN Network Information Center (NIC) will assign you an Internet address, expressed in so-called dotted decimal notation, e.g., `10.33.217.42`, with each byte of the 32-bit Internet address expressed as a decimal number in the range from 0 to 255. This address is the basis of your DDN X.25 address.

DDN X.25 addresses are always either 12 or 14 Binary-Coded Decimal (BCD) digits, depending on whether a subaddress is used. SunNet X.25 does not support subaddressing for X.25 Standard Service, so your `host_address` for Standard Service will always be 12 decimal digits, in the following form:

`0000` *F HHHHHHH*

Where:

`0000`	is used to fill in the DNIC field for a DDN X.25 address.
F	is the flag number. Zero indicates a physical address, 1 a logical address. SunNet X.25 does not implement logical addressing, so this field will always be 0.
HHHHHHH	is the seven-digit host address that you derive from the second and fourth bytes of your Internet address, as described below.

We use the Internet address `10.33.217.42` as an example in the following explanation.

First three digits (PSN number)

The first three digits of the host address are from the last byte of the Internet address. This portion is left-padded with zeroes if the last byte of the Internet address is a one- or two-digit number. For example, `42`, the value of the fourth byte in our example Internet address, would be expressed as `042`. This three-digit number is sometimes referred to as the PSN number.

Next two digits (host port number)

The next two digits are from the second byte of the Internet address, left-padded with a zero if the second byte is a single-digit number. The second byte (`33`) of our example Internet address is appended to `042` to obtain `04233`. This two-digit number is sometimes referred to as the host port number.

Last two digits

The last two digits of the host address are always zeroes.

So, the Internet address `10.33.217.42` becomes the host address `042 33 00`. The complete X.25 address thus becomes:

DNIC	`0000`
Flag	`0`
Host address	`0423300`

When entering your DDN X.25 address for the `address` parameter, divide your address into three sets of digits, separated by hyphens, corresponding to the DNIC, NTN, and subaddress. Thus, you would enter our example address of `0000 0 0423300` as `0000-004233-00`.

D

x25mgr Configuration File

This appendix specifies the parameters and settings for the x25mgr configuration file. It also describes the syntax for the x25mgr and x25admin programs.

The x25mgr program manages virtual circuits for the following uses:

- static virtual circuits for IP over X.25
- dynamic virtual circuits for IP over X.25
- dynamic virtual circuits for IP over DDN X.25 Standard Service
- dynamic virtual circuits for OSI (CLNP) over X.25

NOTE: *The word "static," as we use it here, refers to the fact that the SunNet X.25 always keeps the virtual circuit up. In "dynamic" uses, the X.25 component brings up virtual circuits when there are packets to send and brings them down when there are no packets to send. In this context, "static" and "dynamic" are unrelated to permanent and switched as those words describe virtual circuits. You can use switched or permanent virtual circuits over static links. You can use only switched virtual circuits over dynamic links.*

The parameters for the x25mgr configuration file are explained in the context of each of the above uses.

This material is provided for reference purposes. The install.x25 (or install.osi for SunNet OSI) script creates an x25mgr configuration file that is appropriate for your requirements. See Chapter 4 for background information on IP routing.

D.1. Information that Applies to All Uses

The x25mgr configuration file has the following types of entries:

- configuration commands (iflayer and ifconfig)
- a path section that specifies the calls to originate or accept
- a listener section that specifies calls to listen for
- facilities parameters that specify the desired X.25 facilities (optional for all uses)

The facilities parameters (described in Section D.6) can be in both listener and path sections of the configuration file.

Use hash marks (#) to precede comments in your configuration files. Comments can be on their own line or on the same line with a parameter.

D.2. Configuration File for Static IP/X.25

This section describes the x25mgr configuration file that you use for static IP over X.25 connections.

At the end of this section is an example configuration file for this particular use of SunNet X.25.

Configuration Commands

You must use iflayer to layer IP (ip0) to each xvc device you use. To IP, an xvc device appears as one end of a point-to-point link. The single ip0 can be layered to multiple xvcs. The number of xvcs is configurable in install.x25 (or install.osi). A single xvc corresponds to a single virtual circuit. The following commands layer IP to two xvc devices.

```
shell /usr/sunlink/sync/iflayer ip0 xvc0
shell /usr/sunlink/sync/iflayer ip0 xvc1
```

For each xvc device, you must have an ifconfig command that specifies the local and remote hostnames (or IP addresses) associated with a given xvc device. For example:

```
shell /etc/ifconfig xvc0 unixhost ardennes
shell /etc/ifconfig xvc1 unixhost marseilles
```

As an alternative to the hostnames used in the above example, you can use the IP addresses for each machine. In the ifconfig command, note that the order of the two machines is 1) local host 2) remote host.

If you use subnetting in your network, you must specify the netmask that is used in your network to obtain host addresses. Specify your netmask in the netmask option to ifconfig. For example, if the network represented by ardennes in the previous example uses subnetting and has a netmask of 255.255.255.0, you use the command:

```
shell /etc/ifconfig xvc0 unixhost ardennes netmask 255.255.255.0
```

Listener Section

Below are the parameters for the listener section of the x25mgr configuration file for static IP/X.25. Note that only one listener section is required for multiple IP/X.25 links.

type ip (mandatory)
: Identifies which x25mgr option you are using.

linkid *<number>* (optional)
: If you have multiple links, x25mgr listens on all links, unless you specify a link with this parameter.

`pid cc` (mandatory)
The first four bytes of the Call User Data field in an incoming Call Request. The value `cc` is the pid recommended by RFC 877 to identify IP/X.25 connections. `x25mgr` uses this value to accept (or not accept) incoming calls. This value is used to distinguish between IP/X.25 and other X.25 uses.

`local_x121` <*X.121_address*> (optional)
Normally, `x25mgr` listens on all links for any incoming X.121 address. If you specify this parameter, it should exactly match the contents of the called address field in an Incoming Call packet.

Path Section

The following are the parameters for the path section of the `x25mgr` configuration file for static IP/X.25.

`type ip` (mandatory)
Identifies which `x25mgr` option you are using.

`linkid` <*link_number*> (optional)
Specifies the physical link that `x25mgr` uses to make the call for this path. If you have multiple links, we recommend you do not use this parameter. X.25 automatically selects links, using the algorithm described in Chapter G.

`remote_x121` <*X.121_address*> (mandatory)
The X.121 address of the remote DTE, specified in full, i.e, with DNIC, NTN, and subaddress (if used). If you do not specify `linkid`, above, SunNet X.25 processes this address as specified by the `called_address_opt` and `address_prefix` parameters in the link configuration file for the link automatically selected. See Chapter G for a description of how routing is performed over multiple links. If you specify `linkid`, the address you specify for `remote_x121` must be *exactly* in the form required for the calling or called address fields of Call Request or Incoming Call packets. (The parameters named above are not used to process `remote_x121`.)

`pid cc` (mandatory)
The first part of the Call User Data that `x25mgr` sends in Call Requests. The value here must match the `pid` value in the listener section of the same file.

`local_x121` <*X.121_address*> (optional)
Normally, SunNet X.25 uses the X.121 address in the `address` parameter of the link configuration file. If you specify a value for `local_x121`, SunNet X.25 uses this value instead of the `address` value in a Call Request. If this parameter is used, the address specified must be in *exactly* the form required for the calling address field in a Call Request packet. (If you use `local_x121`, SunNet X.25 does not use the `calling_address_opt` and `address_prefix` parameters in processing the calling address.)

`deadtime` <*number_secs*> (optional)
The maximum amount of time (in seconds) between attempts to establish a connection from the `caller` to the `called` side (see `mode` parameter). The default (and minimum) value for `deadtime` is 10 seconds. Also, the

`deadtime` interval starts at 10 seconds. If you have a `deadtime` value different from the default, `x25mgr` doubles the starting value with each retry until it exceeds the value you entered for `deadtime`, at which point `x25mgr` fixes the `deadtime` interval at the value you entered. You can use the `-enable` option to `x25admin` (see Section D.8) to restart connection attempts at a 10-second interval. When you do this, `x25mgr` restarts the doubling of the call-attempt interval, building toward the interval specified in the `deadtime` parameter.

`interface` <*xvc_number*> (mandatory)
: `xvc` devices are numbered starting at 0.

`mode` *[*`caller|called`*]* (mandatory)
: The string `caller` tells `x25mgr` to make Call Requests. Specifying `called` makes `x25mgr` wait for Incoming Calls.

`name` *path_name* (optional)
: Default is the value of the `interface` parameter. We recommend you use the default, because it is guaranteed to be unique for the path described by a given configuration file. However, you can assign any string to this parameter. If you do assign a value to this parameter, use that value for the `path` argument to the `x25admin` command.

`pvc` *lcn_number* (mandatory for permanent virtual circuits)
: This is the logical channel number for your permanent virtual circuit, as assigned by your PSDN.

Example File

The following file shows the parameters that are mandatory for this use.

```
shell /usr/sunlink/sync/iflayer ip0 xvc0
shell /usr/sunlink/sync/iflayer ip0 xvc1
shell /etc/ifconfig xvc0 unixhost ardennes
shell /etc/ifconfig xvc1 unixhost marseilles

listener        # note listener keyword
type ip
pid cc

path            # first of two paths
type ip
remote_x121 4082701061
pid cc
interface 0     # this refers to xvc0
mode caller

path            # second of two paths
type ip
remote_x121 4153362478
pid cc
interface 1     # this refers to xvc1
mode called
```

D.3. Configuration File for Dynamic IP/X.25

This section describes the `x25mgr` configuration file that you use for dynamic IP over X.25 connections.

At the end of this section is an example configuration file for this particular use of SunNet X.25.

Configuration Commands

You need an `ifconfig` command to assign your local IP address to the `std0` interface and to mark the interface as "up." An `iflayer` command is not required, because `std0` is always associated with `ip0`. The following is an example command as it might appear in your configuration file:

```
shell /etc/ifconfig std0 local_IP_address up
```

Listener Section

Below are the parameters for the listener section of the `x25mgr` configuration file for dynamic IP/X.25.

NOTE: *Only one listener section of type* `ip` *is required for multiple IP/X.25 links, including a dynamic and one or more static links.*

`type ip_dynamic` (mandatory)
: Identifies IP as the recipient for packets received by X.25.

`linkid` <*number*> (optional)
: If you have multiple links, `x25mgr` listens on all links, unless you specify a link with this parameter.

`pid cc` (mandatory)
: The first four bytes of the Call User Data field in an incoming Call Request. The value `cc` is the pid recommended by RFC 877 to identify IP/X.25 connections. `x25mgr` uses this value to accept (or not accept) incoming calls. This value is used to distinguish between IP/X.25 and other X.25 uses.

`local_x121` <*X.121_address*> (optional)
: Normally, `x25mgr` listens on all links for any incoming X.121 address. If you specify this parameter, it should exactly match the contents of the called address field in an Incoming Call packet.

Path Section

The following are the parameters for the path section of the `x25mgr` configuration file for dynamic IP/X.25.

`type ip_dynamic` (mandatory, see note above)
: Identifies the `x25mgr` option you are using.

`linkid` <*link_number*> (optional)
: Specifies the physical link that `x25mgr` uses to make the call for this path. If you have multiple links, we recommend you do not use this parameter. X.25 automatically selects links, using the algorithm described in Chapter G.

`remote_x121` <*X.121_address*> (mandatory)
: The X.121 address of the remote DTE, specified in full, i.e, with DNIC, NTN, and subaddress (if used). If you do not specify `linkid`, above, SunNet X.25 processes this address as specified by the

`called_address_opt` and `address_prefix` parameters in the link configuration file for the link automatically selected. See Chapter G for a description of how routing is performed over multiple links. If you specify `linkid`, the address you specify for `remote_x121` must be *exactly* in the form required for the calling or called address fields of Call Request or Incoming Call packets. (The parameters named above are not used to process `remote_x121`.)

`pid cc` (mandatory)
The Call User Data that `x25mgr` sends in Call Requests. The value here must match the `pid` value in the listener section of the same file.

`local_x121` *<X.121_address>*(optional)
Normally, SunNet X.25 uses the X.121 address in the `address` parameter of the link configuration file. If you specify a value for `local_x121`, SunNet X.25 uses this value instead of the `address` value in a Call Request. If this parameter is used, the address specified must be in *exactly* the form required for the calling address field in a Call Request packet. (If you use `local_x121`, SunNet X.25 does not use the `calling_address_opt` and `address_prefix` parameters in processing the calling address.)

`remote_ip` *remote_IP_address* (mandatory)
Use a hostname, as specified in an `/etc/hosts` file or in the NIS `hosts.byname` database, or an IP address, in standard “dot” notation, e.g., 129.144.133.18. The value for this parameter becomes the `path` argument for the `x25admin` command. Make sure that the hostname or IP address you use identifies a machine that is directly connected (through the PSDN) to your machine.

`timeout` *number_of_seconds* (optional)
This parameter specifies the time that a virtual circuit is kept active before it is closed because of inactivity. The timeout period restarts when a packet is sent or received on the virtual circuit. The default timeout starts at 120 seconds and ranges between 5 and 240 seconds. With decreasing activity, the timeout inches up toward 240; with increasing activity (as manifested by a shortage of file descriptors), the timeout decrements in 20-second chunks. It never falls below 5 seconds. If you change the default, the timeout period you specify remains fixed.

`name` *path_name* (optional)
Default is the value of the `remote_ip` parameter. We recommend you use the default, because it is guaranteed to be unique for the path described by a given configuration file. However, you can assign any string to this parameter. If you do assign a value to this parameter, use that value for the `path` argument to the `x25admin` command.

Example File

The following file shows the parameters that are mandatory for this use.

```
shell /etc/ifconfig std0 local_ip_address up

listener
type ip_dynamic
pid cc

path
type ip_dynamic
remote_ip ardennes  # can also be specified as IP address
remote_x121 2222123456
pid cc
timeout 20

path
type ip_dynamic
remote_ip marseilles
remote_x121 3333654789
pid cc
timeout 10
```

D.4. Configuration File for DDN X.25 (X.25 Standard Service)

This section describes the `x25mgr` configuration file for using X.25 Standard Service for connecting to the DDN.

At the end of this section is an example configuration file for this particular use of SunNet X.25.

Configuration Commands

You need an `ifconfig` command to assign your local Class A or B IP address to the `std0` interface and to mark the interface as "up." An `iflayer` command is not required. The following is an example command as it might appear in your configuration file:

```
shell /etc/ifconfig std0 DDN_IP_address up
```

Listener Section

The following are the parameters for the listener section of the `x25mgr` configuration file for dynamic IP/DDN Standard X.25 Service.

`type ip_std` (mandatory)
: Identifies which `x25mgr` option you are using.

`linkid` <*number*> (optional)
: Even if you have multiple links, `x25mgr` listens on only one link for this use. By default, `x25mgr` listens on link 0. Specify a link only if you want `x25mgr` to listen on a link other than 0.

`pid cc` (mandatory)
: The first four bytes of the Call User Data field in an incoming Call Request. The value `cc` is the pid recommended by RFC 877 to identify IP/X.25 connections. `x25mgr` uses this value to accept (or not accept) incoming calls. This value is used to distinguish between IP/X.25 and other X.25 uses.

Path Section

The following are the parameters for the path section of the `x25mgr` configuration file for dynamic IP/DDN Standard X.25 Service.

`type ip_std` (mandatory)
: Identifies which `x25mgr` option you are using.

`linkid` <*link_number*> (optional)
: Specifies the physical link that `x25mgr` uses to make the call for this path. For this use, `x25mgr` uses link 0 as the default. Use this parameter only if you have multiple links and do not want outgoing calls to go over link 0.

`pid cc` (mandatory)
: The first part of the Call User Data that `x25mgr` sends in Call Requests. The value here must match the `pid` value in the listener section of the same file.

`timeout` *number_of_seconds* (optional)
: This parameter specifies the time that a virtual circuit is kept active before it is closed because of inactivity. The timeout period restarts when a packet is sent or received on the virtual circuit. The default timeout starts at 120 seconds and ranges between 5 and 240 seconds. With decreasing activity, the timeout inches up toward 240; with increasing activity (as manifested by a shortage of file descriptors), the timeout decrements in 20-second chunks. It never falls below 5 seconds. If you change the default, the timeout period you specify remains fixed.

`name` *path_name* (optional)
: Default is the string “std.” We recommend you use the default, because it is guaranteed to be unique for the path described by a given configuration file. However, you can assign any string to this parameter. If you do assign a value to this parameter, use that value for the `path` argument to the `x25admin` command.

Example File

The following file shows the parameters that are mandatory for this use.

```
shell ifconfig std0 interface_ip_address up
# note that interface_ip_address must be Class A or B address

listener
type ip_std
linkid 1    # default is link 0
pid cc

path
type ip_std
linkid 1
pid cc
timeout 20
```

D.5. Configuration File for OSI (CLNP)/X.25

This section describes the `x25mgr` configuration file that you use for the CLNP over X.25 connections.

At the end of this section is an example configuration file for this particular use of SunNet X.25.

Configuration Commands

There are no configuration commands in an `x25mgr` configuration file used for CLNP over X.25. Configuration commands for the SunNet OSI, including CLNP over X.25, are contained in the `/etc/sunlink/osi/rc.config` file.

Listener Section

The following are the parameters for the listener section of the `x25mgr` configuration file for dynamic CLNP/X.25.

`type osi` (mandatory)
Identifies which `x25mgr` option you are using.

`linkid` <*number*> (optional)
If you have multiple links, `x25mgr` listens on all links, unless you specify a link with this parameter.

`pid 81`(mandatory)
The first four bytes of the Call User Data field in an incoming Call Request. The value `81` is a standard value used to identify CLNP/X.25 connections. `x25mgr` uses this value to accept (or not accept) incoming calls. This value is used to distinguish between CLNP/X.25 and IP/X.25 uses.

`local_x121` <*X.121_address*> (optional)
Normally, `x25mgr` listens on all links for any incoming X.121 address. If you specify this parameter, it should exactly match the contents of the called address field in an Incoming Call packet.

Path Section

Below are the parameters for the path section of the `x25mgr` configuration file for dynamic CLNP/X.25. You can have multiple path sections of type `osi`. Each path section must have either a `remote_x121` parameter that is different from the same parameter in other `osi` path sections or not have a `remote_x121` parameter. Only one path section can omit `remote_x121`. If `x25mgr` receives a call request for a destination that is not specified in any `osi` path sections, it uses the values from the path section that does not have a `remote_x121` parameter, if such a section exists.

`type osi` (mandatory)
Identifies which `x25mgr` option you are using.

`linkid` <*number*> (optional)
Specifies the physical link that `x25mgr` uses to make the call for this path. If you have multiple links, we recommend you do not use this parameter. X.25 automatically selects links, using the algorithm described in Chapter G.

`remote_x121` <*X.121_address*> (optional)
The X.121 address of the remote DTE, specified in full, i.e, with DNIC, NTN, and subaddress (if used). If you do not specify `linkid`, above, SunNet X.25 processes this address as specified by the `called_address_opt` and `address_prefix` parameters in the link

configuration file for the link automatically selected. See Chapter G for a description of how routing is performed over multiple links. If you specify `linkid`, the address you specify for `remote_x121` must be *exactly* in the form required for the calling or called address fields of Call Request or Incoming Call packets. (The parameters named above are not used to process `remote_x121`.)

`pid 81` (mandatory)
The first four bytes of the Call User Data field in an outgoing Call Request. The value `81` is a standard value used to identify CLNP/X.25 connections. This value is used to distinguish between CLNP/X.25 and other X.25 uses.

`local_x121` <*X.121_address*> (optional)
Normally, SunNet X.25 uses the X.121 address in the `address` parameter of the link configuration file. If you specify a value for `local_x121`, SunNet X.25 uses this value instead of the `address` value in a Call Request. If this parameter is used, the address specified must be in *exactly* the form required for the calling address field in a Call Request packet. (If you use `local_x121`, SunNet X.25 does not use the `calling_address_opt` and `address_prefix` parameters in processing the calling address.)

`timeout` *number_of_seconds* (optional)
This parameter specifies the time that a virtual circuit is kept active before it is closed because of inactivity. The timeout period restarts when a packet is sent or received on the virtual circuit. The default timeout starts at 120 seconds and ranges between 5 and 240 seconds. With decreasing activity, the timeout inches up toward 240; with increasing activity (as manifested by a shortage of file descriptors), the timeout decrements in 20-second chunks. It never falls below 5 seconds. If you change the default, the timeout period you specify remains fixed.

`name` *path_name* (optional)
Default is the string “osi.” We recommend you use the default, because it is guaranteed to be unique for the path described by a given configuration file. However, you can assign any string to this parameter. If you do assign a value to this parameter, use that value for the `path` argument to the `x25admin` command.

Example File

The following file shows the parameters that are mandatory for this use.

```
listener
type osi
pid 81

path
type osi
pid 81
timeout 20
```

D.6. Specifying Facilities

The facilities parameters apply to all of the X.25 uses described here. These parameters are optional. You can specify all of these parameters for both listener and path sections of the `x25mgr` configuration file. However, the `cug` and `rpoa` have no effect when specified in the listener section.

Table D-1 lists the facilities parameters.

Table D-1 *X.25 Facilities*

Parameter	*Possible Values*	*Description*
`reverse_charge`	0 or 1	In the path section, a 1 setting for this parameter indicates that you can send reverse-charge calls. In the listener section, a 1 setting indicates that you accept reverse-charge calls.
`recvpktsize`	16, 32, 64, 256, 512, 1024, 2048, or 4096	A non-standard maximum packet size for received packets. (Standard and SunNet X.25 default is 128.)
`sendpktsize`	16, 32, 64, 256, 512, 1024, 2048, or 4096	A non-standard maximum packet size for sent packets. (Standard and SunNet X.25 default is 128.)
`rcvwndsize`	1-127	A non-standard window size for received packets. The default is taken from the link configuration file.
`sendwndsize`	1-127	A non-standard window size for sent packets. The default is taken from the link configuration file.
`rcvthruput`	3-12, corresponding to data rates from 75-48000 bps (See the `default_thruput` parameter in section B.4.3 for these values.)	A throughput class that SunNet X.25 will negotiate for received packets. The default is taken from the link configuration file.
`sendthruput`	3-12, corresponding to data rates from 75-48000 bps (See the `default_thruput` parameter in section B.4.3 for these values.)	A throughput class that SunNet X.25 will negotiate for sent packets. The default is taken from the link configuration file.
`cug`	A number determined by the PSDN that identifies a given Closed User Group	A set of DTE users to which your communication will be restricted.
`rpoa`	A number determined by the PSDN that identifies a given RPOA.	The Recognized Private Operating Agency (RPOA) through which a call will be routed.

The following is an example `x25mgr` configuration file (for dynamic IP/X.25) that specifies the `reverse_charge`, `recvthruput`, and `sendthruput` facilities.

```
shell /usr/sunlink/sync/iflayer ip0 std0
shell ifconfig std0 <local ip address> up

listener
type            ip
pid             cc
reverse_charge  1
recvthruput     11  # means throughput class of 19.2 Kbs
sendthruput     11

path
type            ip_dynamic
remote_ip       ardennes
# hostname, above, can also be specified as IP address
remote_x121     2222123456
pid             cc
timeout         20
reverse_charge  1
recvthruput     11  # means throughput class of 19.2 Kbs
sendthruput     11

path
type            ip_dynamic
remote_ip       marseilles
# hostname, above, can also be specified as IP address
remote_x121     3333654789
pid             cc
timeout         10
```

With the above configuration, `x25mgr` accepts reverse-charge calls, because `reverse_charge` is `1` in the listener section. Reverse-charge calls can be sent to the remote host identified in the first path section, where `reverse_charge` is `1`, but not to the host identified in the second path section, where there is no `reverse_charge` parameter. As indicated by the `recvthruput` and `sendthruput` parameters, `x25mgr` negotiates a send and receive throughput class of 11 (19200 bps) for all incoming calls. For all calls to the host identified in the first path section, `x25mgr` also negotiates a send and receive throughput class of 11.

D.7. `x25mgr` Options and Syntax

The `x25mgr` program manages virtual circuits for SunNet X.25. The only required argument to the program is the pathname of a configuration file. `x25mgr` has the following options:

`–v` Specifies verbose mode. With this option in effect, `x25mgr` returns copious status information. Whether `-v` is in effect or not, `x25mgr` directs its status messages to `/var/adm/x25mgrlog`.

`–d` This option causes `x25mgr` to run in debug mode. It does not become a daemon.

–p “Parse-only” mode. This option is useful in checking the syntax in your x25mgr configuration file. It causes x25mgr to exit immediately after it parses the configuration specified in the file.

The install.x25 (or install.osi) script sets up an x25mgr configuration file with the name x25manager.configuration and stores the file in /etc/sunlink/x25.

Following the use of install.x25 (or install.osi), x25mgr is invoked from the startup file /etc/sunlink/x25/rc. In this file, you can comment out the lines that start x25mgr if you want to start the program at run-time instead of boot time.

To start x25mgr at run-time, you enter your command line at the system prompt, just as you would enter any SunOS command. You must be root to invoke x25mgr from the system prompt.

D.8. x25admin Syntax and Options

The x25admin program allows you to enable, disable, or collect statistics for a path or set of paths. Its syntax is:

```
# x25admin [<hostname>] [-enable | -disable | -stats] path
# x25admin -halt
# x25admin -update
```

<*hostname*> is the name of the machine running x25mgr. For all options except -stats, the default for *hostname* is the name of the local machine. The hostname argument is required for the -stats option, described below.

The default value of *path* depends on your use of X.25:

- If you are using IP over static virtual circuits, the default path is the number of the xvc device to which IP is layered.
- For IP over dynamic virtual circuits, the default path is the string used to specify the remote IP address, usually the hostname of a remote host, or the IP address itself, in the traditional Internet “dot” notation, e.g., 129.144.133.18.
- For connections to the DDN, the path defaults to the string “std.”
- For CLNS over X.25, the path defaults to the string “osi.”

You can change the default string by changing the name parameter in the path section of your x25mgr configuration file.

For all of the x25admin options (described below) except -stats, you can enter the path all, to specify all of the paths on a given machine.

The x25admin program has the options described below. -enable, -disable, and -stats are all used in conjunction with a path specifier. The -halt option is used without a path specifier.

`–enable` Allows incoming and outgoing calls on one or all paths.

`–disable` Disallows incoming and outgoing calls on one or all paths. This option brings down virtual circuits at the packet level; HDLC and physical levels remain up.

`–stats` Displays statistics for a single path. Requires that you specify a hostname of a machine running `x25mgr`.

`–update` Stops and restarts `x25mgr`, causing it to reread its configuration file. This option brings down all dynamic virtual circuits. Static IP connections remain up, unless you change the address of one end or the other.

`–halt` Kills the process associated with `x25mgr`.

For all options except `-stats`, you must be `root` on the machine running `x25mgr` to run `x25admin`.

For IP/static links, the `-enable` option, in addition to enabling connections, resets the interval between connection attempts to the starting value of 10 seconds. If the `deadtime` parameter is set in the `x25mgr` configuration file (see Section D.2, use of the `-enable` option restarts the doubling of connection-attempt interval. This doubling stops when `deadtime` value is reached or exceeded; the interval then fixes at value of `deadtime`.

The following `x25admin` command enables connection attempts over `xvc0` and resets the interval between connection attempts to 10 seconds.

```
hostname# x25admin -enable 0
```

The `0` in the command line above indicates `xvc0`, an interface used for IP/static links.

E

Installing SunNet X.25 on a Diskless Workstation

This appendix contains two procedures for installing SunNet X.25 on a diskless machine. Use the first procedure if the client machine and the NFS server are of the same machine architecture (e.g., both Sun-3s). Use the second procedure if the two machines are of different architectures (e.g., a Sun-3 and a Sun-4).

Both procedures assume you are logged in as `root`. These procedures are for Sun-3/-4 and SPARC machines only. In the example commands, `nfs_server` is the name of the NFS server, `client` the name of the diskless machine, and `X25KERNEL` the name of the new kernel. In pathnames, replace *k_arch* with the directory name that indicates a machine's architecture as returned by the `arch -k` command, for example, `sun3x` for a Sun-3/80. Replace *arch* with the name returned by the `arch` command, for example, `sun3` for a Sun-3.

E.1. Procedure 1: Same Architectures

1. On the NFS server, run `extract_unbundled`, as follows:

```
nfs_server# /usr/etc/extract_unbundled
```

   Respond to the prompts as documented in the Chapter 5.

2. On the NFS server, as `root`, make a copy of the file `/etc/exports`. You will later restore this copy. Then, edit `/etc/exports` so that `root` on the diskless machine can mount the `/usr` hierarchy with read/write permissions. To do this, modify the `/usr` line in `/etc/exports` so that it reads:

```
/usr -root=client
```

3. On the NFS server, run `exportfs` to update the NFS exports, as follows:

```
nfs_server# /usr/etc/exportfs -a
```

4. On the diskless machine, remount the `/usr` and `/usr/share` hierarchies as read-write, as follows:

```
client# mount -o remount,rw /usr
client# mount -o remount,rw /usr/share
```

5. On the diskless machine, run `/usr/sunlink/x25/install.x25` to install system files and build a new kernel. Our example kernel is named `X25`.

6. On the diskless machine, move the new kernel to `/vmunix` after saving the existing `vmunix`.

```
client# cd /sys/`arch -k`/X25
client# mv /vmunix /vmunix.old
client# mv vmunix /vmunix
```

7. Reboot the diskless machine, to ensure that the new kernel works correctly.

```
client# /etc/reboot
```

   If you cannot boot the new kernel, follow the procedure in the Chapter 6 to boot single-user.

8. On the NFS server, restore the original `/etc/exports` and rerun `/usr/etc/exportfs -a`.

E.2. Procedure 2: Different Architectures

9. On the NFS server, run `extract_unbundled`, as follows:

```
nfs_server# /usr/etc/extract_unbundled
```

   Respond to the prompts as documented in the Chapter 5.

10. Create the following directories and symbolic link:

```
client# mkdir /export
client# mkdir /export/exec
client# ln -s /usr /export/exec/arch
```

11. On the NFS server, as `root`, make a copy of the file `/etc/exports`. You will later restore this copy. Then, edit `/etc/exports` so that `root` on the diskless machine can mount the `/usr` and `/export/exec` hierarchies with read/write permissions. To do this, modify the `/usr` and `/export/exec` lines in `/etc/exports` so that the lines read:

```
/usr -root=client
/export/exec/arch -root=client
```

12. On the NFS server, run `exportfs` to update the NFS exports, as follows:

```
nfs_server# /usr/etc/exportfs -a
```

13. On the diskless machine, remount the `/usr` and `/usr/share` hierarchies as read-write, as follows:

```
client# mount -o remount,rw /usr
client# mount -o remount,rw /usr/share
```

14. On the diskless machine, run `/usr/sunlink/x25/install.x25` to install system files and build a new kernel. Our example kernel is named `X25`.

15. On the diskless machine, copy the new kernel to `/vmunix` after saving the existing `vmunix`.

```
client# cd /sys/‘arch -k‘/X25
client# mv /vmunix /vmunix.old
client# mv vmunix /vmunix
```

16. Reboot the diskless machine, to ensure that the new kernel works correctly.

```
client# /etc/reboot
```

    If you cannot boot the new kernel, follow the procedure in the Chapter 6 to boot single-user.

17. On the NFS server, restore the original `/etc/exports` and rerun `/usr/etc/exportfs -a`.

F

Cabling

This appendix explains how to build a synchronous null modem and an X.21-to-RS-449 converter.

F.1. Null Modem Requirements

A synchronous null modem is a specially configured cable that simulates two back-to-back modems. When the distance between two host is not great, you might be able to use a null modem instead of more expensive synchronous modems or synchronous modem eliminators. The use of the null modem described here allows you to connect your Sun machine to another machine, Sun or non-Sun, for synchronous communication over a short distance.

The maximum distance at which a null modem can work is determined by the specification for your serial port interface. The RS-232-C specification gives a maximum distance between interfaces (usually machine to modem) of about 50 feet. For RS-449, using the RS-423-A (unbalanced) electrical characteristics, the maximum distance ranges from 4000 feet at 100 Kb/sec., down to 40 feet at the maximum signaling rate of 2 Mb/sec.

In “real life,” users are able to successfully communicate over distances in excess of the distances in the port specifications, especially if they use interference-resistant cabling. However, we cannot make recommendations for between-machines distances beyond the distance specified for your port type.

There are three steps you must perform to use a null modem: for machine-supplied clocking"

- Set the <*mode*> argument to the `x25` parameter in your link configuration file so that one side is `dte` and the other side is `dce`. This step is described in the section “Packet-level Section” in Section B.4.
- Run `syncinit` with certain parameters set so that the Sun machine, in the absence of a synchronous modem, supplies clocking on the serial line.
- Configure the cable for the null modem.

You must `syncinit` each time you reboot your machine.

You can use a null modem with any of Sun’s RS-232-C or RS-449 serial port options.

Configuring Sun Machine to Supply Clocking

To configure an RS-232-C or RS-449 port to supply clocking, set the `txc` (transmit clock) and `rxc` (receive clock) parameters in `syncinit` to `baud` and `rxc`, respectively. For example, the following `syncinit` command, sets the data rate of the first CPU serial port to 9600 bps and sets the clocking as just described:

```
hostname# syncinit zss0 9600 txc=baud rxc=rxc
```

You enter such a command at both ends of a link if boths sides are supplying clocking, or on the Sun that is supplying clocking, if that machine is supplying clocking for both ends of the link.

In the situation in which you have Suns at both ends of a link and have one machine supply clocking for both sides, on the machine that is not supply clocking, you enter:

```
hostname# syncinit zss0 9600 txc=txc rxc=rxc
```

Building the Null Modem

The following subsections provide four null-modem schematic diagrams. The first pair of diagrams are RS-232-C null modems; the second pair are RS-449 null modems. Within each pair, one diagram illustrates a null modem to connect two Sun workstations in which both Suns supply clocking. The other diagram illustrates a null modem to connect a Sun to another machine (Sun or non-Sun), in which one (or, the only) Sun supplies clocking for both sides.

To build a null modem, you can configure your own cable or use a standard cable with an adapter box. If you decide to use an adapter box, be sure to obtain an adapter that allows you to change pin configurations. Pre-configured adapters generally do not work with synchronous protocols, because they do not handle clock signals correctly.

RS-232-C Null Modems

The following diagram illustrates a synchronous null modem that allows you to connect two Sun machines that each supply clocking, using the RS-232-C interface. Each Sun supplies clocking on pin 24. The null modem routes this clocking to pin 17 on the opposite side to provide receive clocking.

Figure F-1 *RS-232-C Null Modem: Both Suns Supply Clocking*

2417

The following diagram illustrates a synchronous null modem that allows you to connect a Sun machine to another machine, Sun or non-Sun, using the RS-232-C interface. The Sun supplies both the transmit and receive clocks for the other machine. Note that this null modem is not symmetrical.

Figure F-2 *RS-232-C Null Modem: Sun Supplies Clocking for Both Sides*

2418

RS-449 Null Modems

The following diagram illustrates a synchronous null modem that allows you to connect two Sun machines that each supply clocking, using the RS-449 interface. Each Sun supplies clocking on pins 17 and 35. The null modem routes this clocking to pins 18 and 26 on the opposite side to provide receive clocking.

Because the RS-449 interface is balanced, there are two pins for each signal. For example, in the diagram below, Transmit Data (TD), pins 4 and 22, is connected to Receive Data (RD), pins 6 and 24. This means that pin 4 is connected to pin 6 and pin 22 is connected to pin 24.

Figure F-3 *RS-449 Null Modem: Both Suns Supply Clocking*

2419

The following diagram illustrates a synchronous null modem that allows you to connect a Sun machine to another machine, Sun or non-Sun, using the RS-449 interface. The Sun supplies both the transmit and receive clocks for the other machine. Note that this null modem is not symmetrical.

Because the RS-449 interface is balanced, there are two pins for each signal. For example, in the diagram below, Transmit Data (TD), pins 4 and 22, is connected to Receive Data (RD), pins 6 and 24. This means that pin 4 is connected to pin 6 and pin 22 is connected to pin 24.

Figure F-4 *RS-449 Null Modem: Sun Supplies Clocking for Both Sides*

2420

F.2. X.21-to-RS-449 Converter

Figure F-5 illustrates the pin connections required for an X.21-to-RS-449 converter.

Figure F-5 *X.21-to-RS-449 Converter*

xxxx

G

X.25 Routing

SunNet X.25 supports routing on full or partial Address Extension Facilities (AEFs) or X.121 addresses. AEFs can be OSI NSAP addresses or non-OSI addresses. This chapter describes the syntax for the SunNet X.25 routing file, which you use for input to the `/usr/sunlink/x25/x25route` command. This chapter also describes the syntax for `x25route`, which you use to add routes from the routing configuration file and perform other routing-related operations.

G.1. Routing File Syntax and Routing Algorithms

The SunNet X.25 routing configuration file is an ASCII text file that you use as input to the `x25route` command (see Section G.2). We suggest that you store your routing file in `/etc/sunlink/x25`, with all of the SunNet X.25 configuration files.

The routing configuration file supports entries for:

- source AEFs†
- AEF routes
- X.121 address routes

For source AEFs, there is a single type of entry, `source_aef`. For AEF and X.121 address routes, there are the following entries:

- `aef_host`
- `aef_prefix`
- `aef_default`
- `x121_host`
- `x121_prefix`
- `x121_default`

These entries are described in detail in the following subsections. The order of these entries is not significant within the file. By convention, source entries precede host entries. Each entry occupies a single line. You can extend an entry to

† You specify source X.121 addresses in the `address` parameter in the link configuration file for each link. The `install.x25` script fills this parameter in for you, in response to your input. See Chapter B for a description of the `address` parameter.

a second line by using the backslash (\) at the end of the line being extended. Precede comments with a hash mark (#) in the first column.

There is a sample routing file, `routes.sample`, stored in `/etc/sunlink/x25`, which contains examples of the routing entries described here.

Source AEFs

SunNet X.25 uses entries for source AEFs if a Call Request packet has a called AEF but not a calling AEF.

Entries for source AEFs have the following form:

```
source_aef <aef_type> <aef_addr>
```

The *<aef_type>* determines the syntax for *<aef_addr>*. The *<aef_type>* is one of:

`osi`
: An OSI NSAP address, expressed in binary (hexadecimal) encoding. This address must be 40 or fewer hexadecimal digits.

`partial-osi`
: An OSI NSAP address with the IDI removed. This is used where the omitted IDI is the same as the called X.121 address. The AFI must be one of the values, 36, 37, 52, or 53, which have IDIs that are X.121 addresses. The partial address must have an even number of digits

`non-osi`
: A non-OSI address. The format of this address is determined by local conventions. This address can have an even or odd number of digits.

The following is an example of a `source_aef` entry with an *<aef_type>* of `osi`:

```
source_aef osi 37000000022255558888888800
```

Note that the above address is a complete NSAP address.

The following is an example of a `source_aef` entry with an *<aef_type>* of `partial-osi`:

```
source_aef partial-osi 378888888800
```

The string of digits in the example above is the NSAP address from the previous example with the IDI digits removed.

The following is an example of a `source_aef` entry with an *<aef_type>* of `non-osi`:

```
source_aef non-osi 8921c6d1
```

Note that the address above is not an OSI address.

You can have multiple `source_aef` entries. If you have multiple entries, SunNet X.25 uses, for a calling address, the `source_aef` entry that most closely matches the destination (called) address. The algorithm for address-matching is as follows:

- SunNet X.25 attempts matches only with compatible entries. Each type (`osi`, `partial-osi`, and `non-osi`) is compatible with itself; the `osi` type is, in addition, compatible with `partial-osi`.
- SunNet X.25 matches sequentially, from left-to-right. It attempts to match byte-by-byte, that is, it does not attempt to match on individual digits or on semi-octets.
- SunNet X.25 uses the longest match it finds. If there are multiple, equal-length matches, SunNet X.25 uses the first match it found.
- If there is no matching `source_aef` entry, SunNet X.25 sends the outgoing packet without a calling AEF.

Consider the following example: SunNet X.25 has an outgoing packet with a destination AEF of 5711111111002222888800, an OSI NSAP address. You have the following `source_aef` entries in your routing configuration file:

```
source_aef osi 5711116666771234888822OO
source_aef partial-osi 5788881100
source_aef partial-osi 57111222200
source_aef non-osi 57111111110022228888
source_aef non-osi abff123
```

SunNet X.25 selects the first entry as a match. Because the destination address is an OSI NSAP address, it considers only the first three entries, ignoring the last two (`non-osi`) entries.

AEF Routes

SunNet X.25 uses AEF routing entries if a Call Request has a called AEF and no called X.121 address. When SunNet X.25 finds a matching entry, it inserts the addresses present in the entry and routes to the link, also specified in the entry.

There are three types of entries associated with AEF routes, `aef_host`, `aef_prefix`, and `aef_default`. SunNet X.25 searches these entries in the order just listed. That is, SunNet X.25 searches:

1. ...for an exact match between the called AEF and the AEF specified in `aef_host` entries.
2. If SunNet X.25 does not find a match among `aef_host` entries, it searches `aef_prefix` entries, starting with the longest entries and working toward the shortest. SunNet X.25 uses the first, longest match it finds.

3. If no match is found among `aef_prefix` entries, SunNet X.25 uses the address(es) specified in the `aef_default` entry. The `aef_default` entry matches any called AEF. You can have only one such entry.

In AEF routing entries, you can specify X.121 and MAC or SNPA addresses. If you supply both an X.121 and a MAC (or SNPA) address, SunNet X.25 inserts both addresses in the Call Request.

`aef_host` Entries

An `aef_host` entry has the following general form:

```
aef_host <aef_addr> <IF_id> [<X.121_addr>] [<MAC_addr>] [<pid>]
```

Following the `aef_host` keyword, the components of an `aef_host` entry are as follows:

<aef_addr>
A complete OSI NSAP address, of 40 or fewer hexadecimal digits.

<IF_id>
Corresponds to the value of the `interface` parameter in your link configuration file. This is the number of an `xpkt` device which is associated with an `hdlc` device, for HDLC links, or, for LLC2 links, an `llc` device.

<X.121_addr>
An X.121 address of 14 or fewer decimal digits that starts with the string `X.121=`. Normally, you do not use an X.121 address if you are connected to a LAN.

<MAC_addr>
Either an 802.*x* SNPA address (MAC address, plus LSEL) or a MAC address. For an SNPA address, precede the address with the string `802=`. For a MAC address, use `MAC=`. MAC addresses are expressed in hexadecimal digits with a colon between each of six bytes. An LSEL (two hexadecimal digits), if present, is preceded by a comma, following the last byte of the MAC address.

If you use a MAC, rather than an SNPA, address, SunNet X.25 appends the LSEL provided in the `lsap` parameter in the link configuration file for the link specified in the *<IF_id>* field of this entry. You do not use a MAC or SNPA address if you are using HDLC links.

<pid>
The protocol id (pid) of no more than five bytes (10 decimal digits). You precede the pid digits with the string `PID=`

The following are example `aef_host` entries:

```
aef_host 37000070755512120000342100 1 X.121=8881217070 MAC=08:00:00:aa:11:11 PID=1234
aef_host 37000000888111100223300 2 802=08:11:22:aa:cc:55,7e
```

Note that the first entry contains an X.121 address, a MAC address, and a pid, while the second address contains only an SNPA address.

`aef_prefix` Entries

An `aef_prefix` entry has the following general form:

```
aef_prefix <aef_prefix> <IF_id> [<X.121_addr>] [<MAC_addr>] [<pid>]
```

Following the `aef_prefix` keyword, the fields in an `aef_prefix` entry are identical to the fields in an `aef_host` entry, except that an `aef_prefix` has an address prefix (*<aef_prefix>*, above) rather than a complete address.

An *<aef_prefix>* is a set of digits that is the leading digits of an AEF.

The following are example `aef_prefix` entries:

```
aef_prefix 37002233 1 X.121=8881217070 MAC=08:00:00:aa:11:11 PID=1234
aef_prefix 370000555 2 802=08:11:22:aa:cc:55,7e
```

Note that the first entry contains an X.121 address, a MAC address, and a pid, while the second address contains only an SNPA address.

`aef_default` Entry

The single `aef_default` entry has the following general form:

```
aef_default <IF_id> [<X.121_addr>] [<MAC_addr>] [<pid>]
```

The `aef_default` entry is the same as the `aef_host` and `aef_prefix` entries, except that an `aef_default` entry does not have an address (*<aef_addr>*) or an address prefix (*<aef_prefix>*).

The following is an example `aef_default` entry:

```
aef_default 1 X.121=8881217070 MAC=08:00:00:aa:11:11 PID=1234
```

In the entry above, note the omission of an address or prefix.

X.121 Address Routes

SunNet X.25 uses X.121 address routing entries if a Call Request has a called X.121 address. When SunNet X.25 finds a matching entry, it inserts the addresses present in the entry and routes to the link specified in the same entry.

There are three types of entries associated with X.121 address routes, `x121_host`, `x121_prefix`, and `x121_default`. SunNet X.25 searches these entries in the order just listed. That is, SunNet X.25 searches:

1. ...for an exact match between the called X.121 address and the X.121 address specified in `x121_host` entries.

2. If SunNet X.25 does not find a match among `x121_host` entries, it searches `x121_prefix` entries, starting with the longest entries and working toward the shortest. SunNet X.25 uses the first, longest match it finds.

3. If no match is found among `x121_prefix` entries, SunNet X.25 uses the address(es) specified in the `x121_default` entry. The `x121_default` entry matches any called X.121 address. You can have only one such entry.

4. If no routing entries match (indicating there is no `x121_default` entry in the routing file), SunNet X.25 checks the DNIC in the `address` parameter in each link configuration file, looking for a match with the DNIC in the called X.121 address. If it finds a match, SunNet X.25 routes the call to the link specified in the `interface` parameter in the link configuration file where the DNIC match was found.

5. If no DNIC match is found and link 0 is not a LAN interface, SunNet X.25 routes the call to link 0.

In X.121 address routing entries, you can specify MAC or SNPA addresses if the routing entry specifies a LAN interface.

`x121_host` Entries

An `x121_host` entry has the following general form:

```
x121_host <X.121_addr> <IF_id> [<MAC_addr>]
```

Following the `x121_host` keyword, the components of an `x121_host` entry are as follows:

<X.121_addr>
: A complete X.121 address, of 14 or fewer decimal digits. You do not precede the X.121 address with the string `X.121=`.

<IF_id>
: Corresponds to the value of the `interface` parameter in your link configuration file. This is the number of an `xpkt` device which is associated with an `hdlc` device, for HDLC links, or, for LLC2 links, an `llc` device.

<MAC_addr>
: Either an 802.*x* SNPA address (MAC address, plus LSEL) or a MAC address. For an SNPA address, precede the address with the string `802=`. For a MAC address, use `MAC=`. MAC addresses are expressed in hexadecimal digits with a colon between each of six bytes. An LSEL (two hexadecimal digits), if present, is preceded by a comma, following the last byte of the MAC address.

  If you use a MAC, rather than an SNPA, address, SunNet X.25 appends the LSEL provided in the `lsap` parameter in the link configuration file for the link specified in the *<IF_id>* field of this entry. You do not use a MAC or SNPA address if you are using HDLC links.

The following are example `x121_host` entries:

```
x121_host 7075551212 1 MAC=08:00:00:aa:11:11
x121_host 8881111002233 2
```

Note that the first entry contains an X.121 address and a MAC address, while the second address contains only an X.121 address.

`x121_prefix` Entries

An `x121_prefix` entry has the following general form:

```
x121_prefix <x121_prefix> <IF_id> [<MAC_addr>]
```

Following the `x121_prefix` keyword, the fields in an `x121_prefix` entry are identical to the fields in an `x121_host` entry, except that an `x121_prefix` has an X.121 address prefix (*<x121_prefix>*, above) rather than a complete X.121 address. This prefix is *not* preceded by the string `X.121=`.

An *<x121_prefix>* is a set of digits that is the leading digits of an X.121 address. You do not precede the X.121 address prefix with the string `X.121=`.

The following are example `x121_prefix` entries:

```
x121_prefix 8881217070 1 MAC=08:00:00:aa:11:11
x121_prefix 12340000555 2
```

Note that the first entry contains an interface id and a MAC address, while the second address contains only the interface id.

`x121_default` Entry

The single `x121_default` entry has the following general form:

```
x121_default <IF_id> [<MAC_addr>]
```

The `x121_default` entry is the same as the `x121_host` and `x121_prefix` entries, except that an `x121_default` entry does not have an X.121 address (*<X.121_addr>*) or an X.121 address prefix (*<X.121_prefix>*).

The following is an example `x121_default` entry:

```
x121_default 1 MAC=08:00:00:aa:11:11
```

In the entry above, note the omission of an X.121 address or prefix. Also, the inclusion of a MAC or SNPA address is optional.

G.2. The `x25route` Command

Use the `x25route` command to add the routes specified in your routing configuration file. Included in these routes are any `source_aef` entries you might have in the file. `x25route` can also display run-time routing configuration information.

The command has the following syntax:

```
# /usr/sunlink/x25/x25route -s [<interval>]
# /usr/sunlink/x25/x25route [-f] [<routing_file>]
```

The options and arguments to this command are:

`-s` Show all routes, including number of packets using each entry. An example of this option is shown below.

<interval>
An interval (number of seconds) at which `x25route` updates its display when you invoke the `-s` option. This is useful for observing the degree of use of a particular routing entry.

`-f` Flush all routes. You can use this option in combination with a routing file. With this combination, `x25route` flushes all routes, *then* adds the routes from the routing file.

<routing_file>
The full pathname of a routing configuration file, as that file is described in Section G.1. When invoked with this argument, `x25route` adds the routes contained in the specified file.

The following example shows an example routing file and the resulting output from a `x25route -s` command.

```
hostname% cat /etc/sunlink/x25/routes

# sample X.25 routing file

source_aef osi          37000012988841417777700
source_aef partial-osi  360000
source_aef non-osi      876a7fc9867e0

aef_host    571111404238510000  0
aef_host    370000404238510000  0 X.121=2345654367  MAC=08:00:20:12:34:56 PID=1234
aef_host    37000040423851000abcdef00   1           802=08:00:20:78:90:12,7e

x121_host   311067895401    0
x121_host   6789543267      1   802=1:2:3:4:5:6,7e
```

Tabs are included above only to improve readability.

```
hostname# x25route -f /etc/sunlink/x25/routes
hostname% x25route -s
Type Host/prefix          Link X.121          MAC                    PID       Use
XH   311067895401            0                                                   0
XH   6789543267              1                01:02:03:04:05:06,7e             300
AH   571111404238510000      0                                                   0
AH   370000404238510000      0 2345654367     08:00:20:12:34:56     1234         0
AH   37000040423851000abcdef00
                             1                08:00:20:78:90:12,7e               0
SO   370000129888841417777                                                     164
SP   360000                                                                      0
SN   876a7fc9867e0
hostname%
```

The fields in the `x25route` display are as follows:

`Type`
: Indicates the type of routing file entry. These entries are described in Section G.1. The letters in the `Type` field are defined as follows:

- `SO` source OSI address entry
- `SP` source partial-OSI address entry
- `SN` source non-OSI address entry
- `A*` AEF entry
- `X*` X.121 address entry
- `*H` host entry
- `*P` prefix entry
- `*D` default entry

`Host/prefix`
: Contains the AEF, or the X.121 or non-OSI address or address prefix.

`Link`
: Contains the link number. This corresponds to the interface id for a routing file entry.

`X.121`
: Contains an X.121 address, if present for a particular entry.

`MAC`
: Contains a MAC or SNPA address, if present for a particular entry. (MAC addresses have six bytes; SNPA addresses have a seventh byte following a comma.)

`PID`
: Contains a protocol id, if present for a particular entry.

`Count`
: Displays the number of packets for which SunNet X.25 used a given entry.

H

X.121 Country Codes

This appendix lists the country codes from Annex D of Recommendation X.121 of the . These are three-digit codes in the range 200 to 799. The first digit of the code indicates the geographical region:

Code Range	Geographical Region
200-299	Europe
300-399	North America
400-499	Asia and the Middle East
500-599	South Pacific
600-699	Africa
700-799	South America

These three-digit codes are referred to as the data country code (DCC). When this code is followed by a fourth digit, it forms the data network identification code (DNIC). The fourth digit identifies a particular public data network within a country. Up to ten different networks can be associated with a single DCC. If a country or region has more than ten networks, then it must use additional DCCs. For example, the United States has been assigned the codes 310 through 316, inclusive.

Code	Country or Geographical Area
202	Greece
204	Netherlands
206	Belgium
208	France
212	Monaco
214	Spain
216	Hungary
218	German Democratic Republic
220	Yugoslavia
222	Italy
226	Romania
228	Switzerland
230	Czechoslovakia
232	Austria
234-235	United Kingdom
238	Denmark
240	Sweden

Code	Country or Geographical Area
242	Norway
244	Finland
250	Union of Soviet Socialist Republics
260	Poland
262	Federal Republic of Germany
266	Gibraltar
268	Portugal
270	Luxembourg
272	Ireland
274	Iceland
276	Albania
278	Malta
280	Cyprus
284	Bulgaria
286	Turkey
288	Faroe Islands
290	Greenland
302	Canada
308	St. Pierre and Miquelon
310-316	United States
330	Puerto Rico
332	Virgin Islands
334	Mexico
338	Jamaica
340	French Antilles
342	Barbados
344	Antigua and Barbuda
346	Cayman Islands
348	British Virgin Islands
350	Bermuda
352	Grenada
354	Montserrat
356	St. Kitts
358	St. Lucia
360	St. Vincent and the Grenadines
362	Netherlands Antilles
364	Bahamas
366	Dominica
368	Cuba
370	Dominican Republic
372	Haiti
374	Trinidad and Tobago
376	Turks and Calcos Islands
404	India
410	Pakistan
412	Afghanistan
413	Sri Lanka
414	Burma
415	Lebanon
416	Jordan
417	Syrian Arab Republic
418	Iraq
419	Kuwait
420	Saudi Arabia

Code	Country or Geographical Area
421	Yemen (Arab Republic)
422	Oman
423	Yemen (People's Democratic Republic of)
424	United Arab Emirates
425	Israel
426	Bahrain
427	Qatar
428	Mongolian People's Republic
429	Nepal
430	United Arab Emirates (Abu Dhabi)
431	United Arab Emirates (Dubai)
432	Iran
440	Japan
450	Korea (Republic of)
452	Viet Nam
454	Hong Kong
455	Macao
456	Democratic Kampuchea
457	Lao People's Democratic Republic
460	China (People's Republic of)
467	Korea (Democratic People's Republic of)
470	Bangladesh
472	Maldives
502	Malaysia
505	Australia
510	Indonesia
515	Philippines
520	Thailand
525	Singapore
528	Brunei
530	New Zealand
535	Guam
536	Nauru
537	Papua New Guinea
539	Tonga
540	Solomon Islands
541	New Hebrides
542	Fiji
543	Wallis and Futuna Islands
544	American Samoa
545	Gilbert & Ellice Islands
546	New Caledonia and Dependencies
547	French Polynesia
548	Cook Islands
549	Western Samoa
602	Egypt
603	Algeria
604	Morocco
605	Tunisia
606	Libya
607	Gambia
608	Senegal
609	Mauritania
610	Mali

Code	Country or Geographical Area
611	Guinea
612	Ivory Coast
613	Burkina Faso
614	Niger (Republic of the)
615	Togolese Republic
616	Benin
617	Mauritius
618	Liberia
619	Sierra Leone
620	Ghana
621	Nigeria (Federal Republic of)
622	Chad
623	Central African Republic
624	Cameroon
625	Cape Verde
626	Sao Tome and Principe
627	Equatorial Guinea
628	Gabon Republic
629	Congo
630	Zaire
631	Angola
632	Guinea-Bissau
633	Seychelles
634	Sudan
635	Rwanda
636	Ethiopia
637	Somali Democratic Republic
638	Republic of Djibouti
639	Kenya
640	Tanzania
641	Uganda
642	Burundi
643	Mozambique
645	Zambia
646	Madagascar
647	Reunion (French Department of)
648	Zimbabwe
649	Namibia
650	Malawi
651	Lesotho
652	Botswana
653	Swaziland
654	Comoros
655	South Africa
702	Belize
704	Guatemala
706	El Salvador
708	Honduras
710	Nicaragua
712	Costa Rica
714	Panama
716	Peru
722	Argentine Republic
724	Brazil
730	Chile

Code	Country or Geographical Area
732	Colombia
734	Venezuela
736	Bolivia
738	Guyana
740	Ecuador
742	Guiana (French Department of)
744	Paraguay
746	Suriname
748	Uruguay

I

Acronyms

This appendix lists the acronyms used in this manual, with their definitions.

ACK
: short for ACKnowledgement

AD
: short for ADdendum

AEF
: Address Extension Facility

AFI
: Authority and Format Identifier

ANSI
: American National Standards Institute

AFI
: Authority and Format Identifier

ANSI
: American National Standards Institute

ARPA
: Advanced Projects Research Agency (now DARPA)

ARPANET
: ARPA NETwork

BCD
: Binary-Coded Decimal

CCITT
: International Telephone and Telegraph Consultative Committee (acronym is French)

CLNP
: Connection-Less Network Protocol

CLNS
: Connection-Less Network Service

CONS
: Connection-Oriented Network Service

COTS
: Connection-Oriented Transport Service

CRC
: Cyclical Redundancy Check

CSU
: Channel Service Unit

CUG
: Closed User Group

DA
: Default Address

DAD
: Draft Addendum

DARPA
: Defense Advanced Projects Research Agency

DCC
: Data Country Code

DCE
: Data Circuit-terminating Equipment or Data Communication Equipment

DDN
: Defense Data Network

DIS
: Draft International Standard

DNIC
: Data Network Identification Code

DSE
: Directory Service Entity

DSP
: Domain-Specific Part

DTE
: Data Terminal Equipment

EGP
: Exterior Gateway Protocol

ER Error Report

ES End System

ESH
: End System Hello

FDDI
: Fiber Distributed Data Interface

FIPS
Federal Information Processing Standard

FTAM
File Transfer, Access, and Management

GOSIP
Government Opens Systems Interconnection Profile

HDH
HDLC Distant Host (Protocol)

HDLC
High-level Data Link Control

HSI
High-speed Serial Interface

ICD
International Code Designator

ICMP
Internet Control Message Protocol

IDI
Initial Domain Identifier

IDP
Initial Domain Part

IEEE
Institute of Electrical and Electronic Engineers

IGP
Interior Gateway Protocol

IMP
Interface Message Processor

IP Internet Protocol

IS Intermediate System or International Standard

ISDN
Integrated Services Digital Network

ISH
Intermediate System Hello

ISO
International Standards Organization

ISODE
ISO Development Environment

IWU
Interworking Unit

LAN
Local-Area Network

LAP
Link Access Protocol

LAPB
Link Access Protocol Balanced

LCN
Logical Channel Number

LLC
Logical Link Control

LLC1
Logical Link Control, Type 1

LLC2
Logical Link Control, Type 2

LSAP
Link Service Access Point

LSEL
Link Selector

MAC
Medium Access Control

MAP
Manufacturers Automation Protocols

MCP
Multi-protocol Communication Processor

MHS
Message-Handling System

MILNET
MILitary NETwork

MTA
Message Transfer Agent

NBS
National Bureau of Standards (renamed National Institute of Standards and Technology)

NFS
Network File System

NIC
Network Information Center

NIS
Network Information Service

NIST
National Institute of Standards and Technology

NOC
Network Operations Center

NSAP
Network Service Access Point

NSEL
Network Selector

NTN
Network Terminal Number

OA
Optional Address

OSI
Open Systems Interconnection

OSINET
(not an acronym) Network for OSI implementations, run by NIST

PAD
Packet Assembler/Disassembler

PDN
Public Data Network

PDU
Protocol Data Unit

PID
Process IDentifier

PLP
Packet Layer Protocol

PSAP
Presentation Service Access Point

PSDN
Packet-Switched Data Network

PSEL
Presentation Selector

PSN
Packet-Switched Node

PSTN
Public Switched Telephone Network

PTT
Public Telephone and Telegraph

PVC
Permanent Virtual Circuit

QC
Query Configuration

RD
Receive Data

RFC
Request for Comment

RIP
Routing Information Protocol

RNR
Receiver Not Ready

RPOA
Recognized Private Operating Agency

RR Receiver Ready

SA Source Address

SABM
Set Asynchronous Balanced Mode

SAL
Source Address List

SAP
Service Access Point

SEL
SELector

SNAP
Subnetwork Access Protocol

SNPA
Subnetwork Point of Attachment

SPARC
Scalable Processor ARChitecture

SRI
Stanford Research Institute

SSAP
Session Service Access Point

SSEL
Session Selector

SVC
Switched Virtual Circuit

TCP
Transmission Control Protocol

TD Transmit Detect

TOP
Technical and Office Protocols

TP Transport Protocol or Transaction Program

TP0
Transport Protocol Class 0

TP2
Transport Protocol Class 2

TP4
Transport Protocol Class 4

TSAP
Transport Service Access Point

TSEL
Transport Selector

UA
Unnumbered Acknowedgement

VC Virtual Circuit

WAN
Wide-Area Network

J

Glossary

CCITT — Consultative Committee for International Telephony and Telegraphy
An international organization of communication carriers, most of which are government telephone and telegraph agencies. The CCITT develops telecommunication standards through the use of their recommendations. The X.25, X.3, and X.29 standards originated as CCITT recommendations.

CUG — Closed User Group
A closed user group permits the DTEs belonging to the group to communicate with each other, but precludes communication with other DTEs. A single DTE can belong to multiple CUGs.

DCE — Data Circuit-terminating Equipment
The network side of the user-to-network interface.

DNIC — Data Network Identification Code
A (usually) four-digit number that identifies a specific PSDN. The DNIC is the first component of a complete X.121 address and is comparable to the exchange portion of a telephone number used for switched telephone service.

DTE — Data Terminal Equipment
The device at the user's side of a user-to-network interface. This may be a computer system (packet-mode DTE) or a character-mode terminal (DTE-C), and is both a source and destination for data.

DTE-C — Data Terminal Equipment-Character Mode
A character-mode (asynchronous) terminal that uses a PAD to connect to a remote host across a PSDN.

NTN — Network Terminal Number
The component of a complete X.121 address that identifies a specific DTE.

PAD — Packet Assembler/Disassembler
An interface device that buffers data sent to or from character mode devices, and assembles and disassembles the packets needed for X.25 operation.

PDN — Public Data Network
: A data communications network whose services are available to any user willing to pay for them. Also used to mean an X.25-compatible Private Data Network. PDN is not used in this manual.

PSDN — Packet-Switched Data Network
: A more general term than PDN, refers to any public or private packet-switching network that provides X.25-compliant interfaces to its users.

PVC — Permanent Virtual Circuit
: A permanent logical association between two physically separate DTEs that does not require call set-up or clearing procedures.

RPOA — Recognized Private Operating Agency
: An X.25 user facility that provides for user specification of a particular RPOA transit network through which a call is to be routed internationally when more than one RPOA transit network exists at an international gateway.

subaddress
: The component of an X.121 address that identifies a specific application on a DTE. The subaddress is an optional component of a complete X.121 address.

SVC — Switched Virtual Circuit
: A temporary logical association between two physically separate DTEs that exists only for the duration of the data transfer. Call set-up and call clearing procedures are required with a switched virtual circuit.

X.21bis
: A set of CCITT recommendations that define the physical interface between a DTE and a DCE of a public data network. Access to the DCE is through synchronous modems and voice-band lines. Equivalent to RS-232-C and V.24/V.28.

X.25 client
: An application layered above X.25.

X.25 gateway
: A Sun workstation on which SunLink X.25 is installed and made available to remote terminal or workstation users.

X.29
: Defines the interface between PADs and packet-mode DTEs or other PADs.

X.3
: Describes the functions of the PAD and the various parameters used to specify its mode of operation.

K

gated Man Pages

This appendix contains the `man` pages for the `gated` (pronounced GAIT-DEE) routing daemon, a software package made available by the Center for Theory and Simulation in Science and Engineering of Cornell University to all Internet (including DDN) users at no cost.

All `gated`-related files—including source code, `makefile`, README files, example configuration files, and `man` pages—are stored in or under `/usr/sunlink/x25/gated`.

We provide `gated` as a convenience to you. This software is supported by the people listed in the `gated/README` file and at the end of the `man` pages. It is not supported by Sun.

NOTE: *If you run the RIP (Routing Information Protocol) portion of* `gated`, *do not run the SunOS-provided* `routed`.