IRIS OSI Transport Service Administration Guide

Document Number 007-1547-020

Contributors

Production: Lorrie Williams

IRIS OSI Transport Service Administration Guide
Document Number 007-1547-020

Silicon Graphics, Inc.
Mountain View, California

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

PREFACE

The IRIS OSI Transport Service software package is part of the Silicon Graphics (SGI) series of IRIS networking products. These products are designed to merge the power and flexibility of the IRIS networking system with the internationally–standardized ISO–OSI communications protocols. The IRIS OSI Transport Service consists of two modules:

- the LT–610 LAN Transport module
- the WT–325 WAN Transport module

The LT–610 LAN Transport module provides Transport Class 4, Connectionless Network Protocol (CLNP), End System to Intermediate System (ES-IS) routing, and Logical Link Control (LLC1) for all IRIS OSI applications, and for any other applications that use the standard AT&T System V Transport Layer Interface (TLI). A SubNetwork Dependence Convergence Function (SNDCF) is included for interconnecting with the WT–325 module.

The WT–325 WAN Transport module provides Transport Classes 0/2/4 for all IRIS OSI applications, and for any other applications that use these communication services.

The IRIX OSI Transport Service documentation package consists of the following documentation:

- This manual (The *IRIS OSI Transport Service Administration Guide*)
- The *IRIS OSI Transport Service Release Notes* (including Installation Instructions)
- The *IRIS OSI Network Directory Compiler Guide*

Intended Audience

This guide is intended for administrators of IRIS–based OSI networks. It assumes a thorough understanding of:

- the IRIS operating system
- the STREAMS networking system
- terms and concepts associated with OSI networks
- LAN and/or WAN networking protocols

A glossary is included that defines key communication terms and acronyms used in this manual.

Contents of This Manual

This guide contains the following sections:

- **Chapter 1** introduces the general context of OSI transport services and the STREAMS networking environment describing the main components of the LT–610 and WT–325 modules.
- **Chapter 2** describes how to select the proper profile for your particular network configuration. Your configuration may require use of either the WT–325 and LT–610 products, or both.
- **Chapter 3** describes how to configure LT–610 and WT–325 in the STREAMS environment.
- **Chapter 4** describes how to invoke the LT–610 network management utilities and how to interpret their output.
- **Chapter 5** describes how to invoke the WT–325 network management utilities and how to interpret their output.
- **Appendix A** describes how to test your LT–610 and WT–325 installation using the **`strxt`** and **`sttxt`** programs.
- **Appendix B** contains an example network description for the Network Directory Compiler (NDC).
- **Appendix C** lists references and standards documents that you may find useful in learning about OSI protocols and UNIX STREAMS concepts.

A **Glossary** defines important terms and acronyms used throughout the manual.

Recommended Additional Reading

Information on using IRIX STREAMS can be found in the following IRIX manuals:

- *STREAMS Primer* (Part Number 007-0831-030)
- *STREAMS Programmer Guide* (Part Number 007-0833-030)

It is also recommended that programmers obtain a copy of the AT&T (UNIX Systems Laboratories, Inc.) guide to programming with the TLI, *UNIX® SYSTEM V RELEASE 4 Programmer's Guide: Networking Interfaces*

Protocol Profiles Supported

The protocol stack provided in the LT–610 module is fully conformant with the protocol profiles developed by the following standards organizations:

- US GOSIP 1.1
- UK GOSIP 3.0
- COS 1.1
- TOP 3.0
- NIST SP 500-162

The protocol stack provided in the WT–325 module is fully conformant with the protocol profiles developed by the following standards organizations:

- US GOSIP 1.1
- UK GOSIP 3.0
- NIST SP 500-162
- COS 1.1
- SPAG
- CEN/CENELEC
- TOP 3.0
- MAP 3.0
- Public X.400 services

For a detailed list of the standards and documents relevant to IRIX OSI Transport Services, see Appendix C.

Conventions Used in This Manual

The following typographical conventions are used in this manual:

- All user input appears in the **`Bold Courier`** font. Screen output appears in `Plain Courier` font. The following example shows a login prompt displayed on the screen, followed by a user response:

  ```
  login: smith
  ```

- Command names are shown in **`Bold Courier`**. For example:

  Invoke the **`osid`** STREAMS configuration utility.
  `Enter:`**`osid`**

- File names are shown in `Courier`. For example:

  Modify the `etc/osid.cfg` file.

- When you must choose a particular item or value in a class of items or values, the class is printed in *italics*. For example, if the command were:

  `rtl <`*`file`*`>`

  you should substitute an appropriate filename in place of **`file`**.

- When you have an option to either enter or not enter a particular portion of a command, the optional portion is enclosed in square brackets. For example:

  `net [title]`

- An ellipsis following an optional argument indicates that the command allows you to enter one or more additional arguments. For example:

  `addnsap [NSAP_address] ...`

- When special keyboard keys are mentioned, they are enclosed in angle brackets. For example:

  ```
  <Esc>
  <Ctrl>
  <Del>
  <Return>
  ```

- This manual does not instruct you to press the `<Return>` key after entering a command. Assume that you press `<Return>` after each command unless you are instructed not to do so.

1 . INTRODUCTION

This chapter presents an overview of the IRIS OSI Transport Service software modules.

Contents

1.1. OSI Transport Services

An open system can communicate and interoperate with other open systems from different vendors by using common rules or conventions called protocols. The computer industry is moving away from using proprietary protocols and evolving toward the international Open Systems Interconnection (OSI) international protocols.

IRIS OSI Transport Service is designed to enable the IRIS user to utilize the platform-independent connectivity supplied by the increasingly widespread use of OSI protocols. The LT–610 module provides OSI LAN Transport Services, while WT–325 provides the WAN equivalent. SNDCF (SubNetwork Dependence Convergence Function) is used to enable both LAN and WAN facilities to be simultaneously and transparently available to the user.

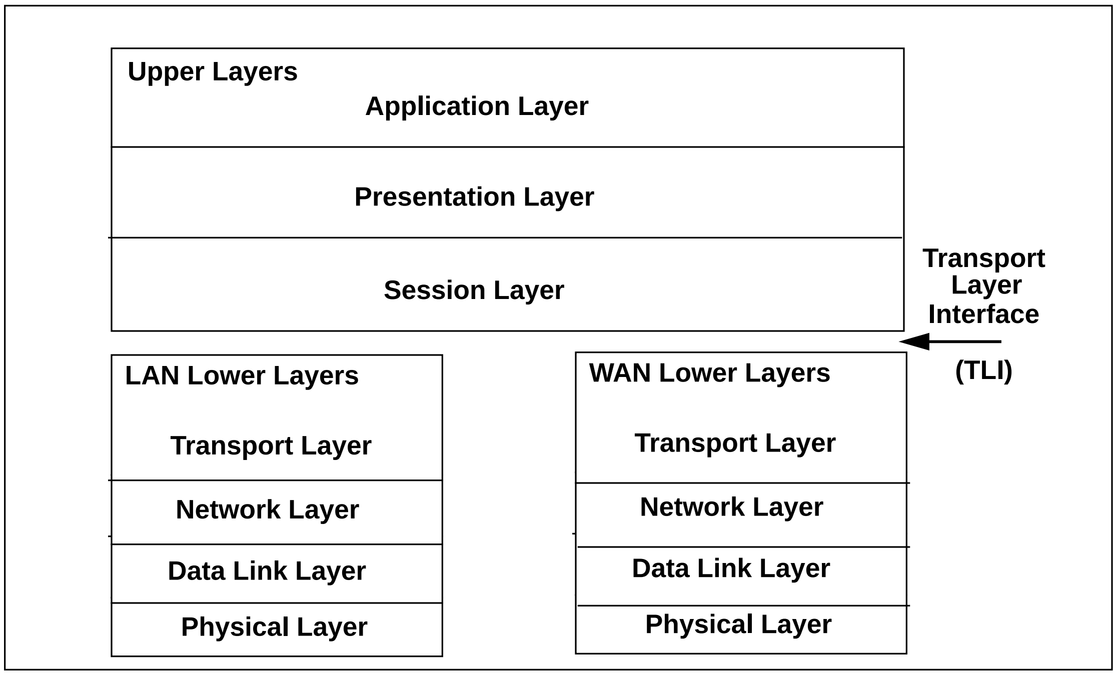


Figure 1-1. The OSI Stack

A distinction is often made between OSI *upper layers*, which deal with issues of relevance to the application or human network user, and the *lower layers*, which deal with network issues such as collision control or routing.

As can be seen in Figure 1-1, OSI Upper Layers operate independently of the distinction between LAN and WAN. These upper layers use a well-defined Transport Layer interface (TLI) to call on OSI Transport Services. This interface is provided by both IRIS Transport Service modules.

1.2. Product Overview

The IRIS OSI Transport Service consists of the following components:

- LT–610 Transport Service software module, which provides OSI LAN Transport plus SNDCF functionality.
- WT–325 Transport Service software module, which provides OSI WAN Transport.
- The Network Directory Compiler, which can be used to generate routing tables for each router (intermediate system) and end system in a network, from a description of the network topology written in a text NDC source file.
- This manual, the *IRIS OSI Transport Service Administration Guide*, which describes the operation of LT–610 and WT–325.
- The *IRIS OSI Transport Service Network Directory Compiler Guide*, which describes the operation of the Network Directory Compiler.
- The *IRIS OSI Transport Service Release Notes.*

1.3. STREAMS Overview

Another concept central to the understanding of IRIS OSI Transport Service is that of STREAMS.

Input and output facilities provided in older versions of the UNIX operating system were mostly character–oriented; they processed one character at a time in a data stream. STREAMS provides the first industry–standard set of UNIX operating system tools optimized to process groups of characters (“messages” in STREAMS terminology). It provides the facilities to efficiently implement modern networking protocols.

The structure of a general protocol stack developed under STREAMS guidelines is shown in Figure 1-2 below. The stack resides within the operating system kernel. It contains a Stream head (which communicates with a user application), zero or more STREAMS modules or multiplexing pseudo device drivers, and a physical device driver. STREAMS modules and drivers communicate with each other through queues. Each module or driver has a read queue for receiving messages that arrive upstream, and a write queue for receiving messages moving downstream.

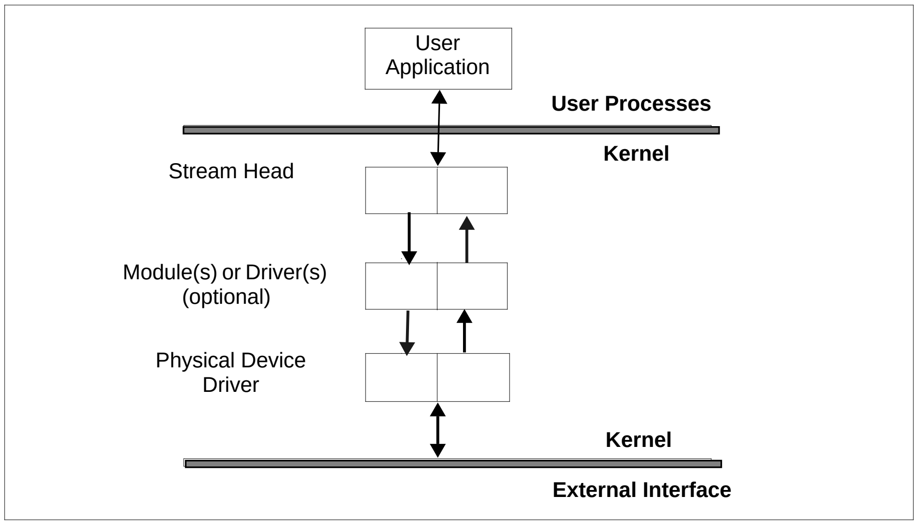


Figure 1-2. Configuration of a General STREAMS Protocol Stack

1.4. Introducing LT–610

The LT–610 LAN Transport Service module implements the service standards and protocols for the OSI transport, network, and data link layers. The software modules, called drivers, are layered into the IRIX STREAMS environment as a protocol stack over the media access control (MAC) device driver. With each driver performing specific functions, the OSI stack supports applications that require data transfer and routing services across a network or interconnected networks.

The LT–610 software includes the following types of drivers:

- STREAMS multiplexing drivers, each designed to provide a set of services and to process data from multiple sources
- A timer driver that performs timer management routines

The LT–610 software requires a compatible MAC driver to be present in the IRIX kernel.

The rest of this section introduces the following topics:

- LT–610 protocols
- LT–610 utility programs for LAN management
- LT–610 configuration when used with WT-325

1.4.1. LT–610 Transport Software

The LT–610 product implements the lower layers of the seven–layer OSI model shown in Figure 1-3a. The shaded boxes represent the layers provided in the LT–610 package. Figure 1-3b shows the LT–610 drivers configured as an OSI protocol stack in the STREAMS environment to support applications that use System V Transport Layer Interface (TLI).

Transport and Network Service Providers

LT–610's transport and network functions reside in one multiplexing driver. The OSI connection-oriented transport protocol, class 4 (TP4), operates over OSI connectionless network service (CLNS).

The network layer portion of the driver implements the OSI Connectionless Network Protocol (CLNP), also known as Internetwork Protocol (IP). The end system to intermediate system routing exchange protocol (ES–IS) used in conjunction with CLNP provides a dynamic exchange of routing information between end systems and intermediate systems. (If the LAN you are accessing does not use this protocol, you can create static routing tables using the Network Directory Compiler, and add them using the network management utility `addnsap` described in section 4.1, on page 4-2.)

Data Link Service Provider and MAC Device Driver

The data link layer functions are structured into two sublayers: Logical Link Control (LLC) and media access control (MAC). The LT–610 product provides connectionless services to the network layer using the protocol Logical Link Control, type 1 (LLC1). The LLC1 driver interfaces to a MAC driver (`/dev/snif`) residing in the IRIX kernel. This driver is not part of the LT–610 product: it simply provides the MAC layer interface needed by the LLC1 driver. Section 3.2, on page 3-2, discusses the entries for these drivers in the configuration file.

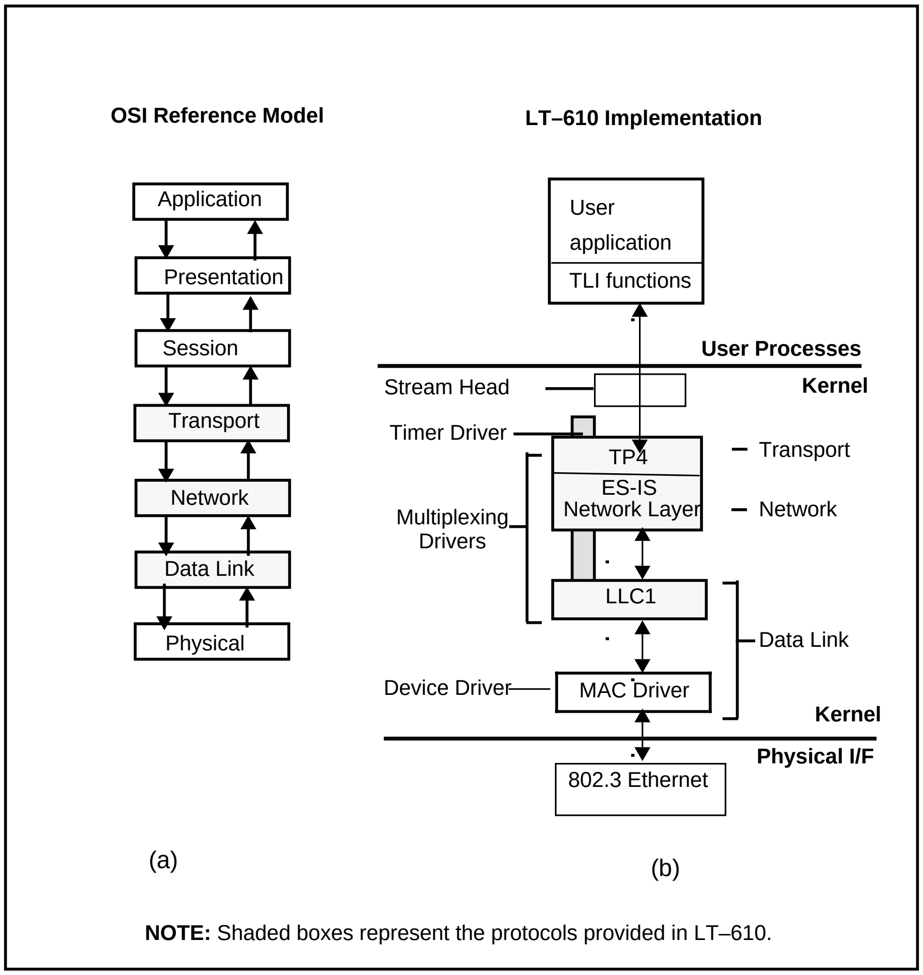


Figure 1-3. LT–610's OSI Protocol Stack

1.4.2. LT–610 Utility Programs

Table 1-1 lists the utility programs that are used to manipulate various aspects of a LAN protocol stack. Chapter 4 discusses each in detail.

Table 1-1. LAN Management Utilities Grouped by Layers and Protocols

Utility	Description
Transport	
`tp4stat`	Displays the management statistics that TP4 maintains.
`tp4config`	Configures various Transport Layer counters, timers, and data unit formats. Parameter values are included as command line options.
Network, IP	
`ipconfig`	Configures various IP layer counters, timers, and data unit formats. Parameter values are included as command line options.
`ipstat`	Displays the management statistics that IP maintains.
`net`	Loads or displays the local IP network entity title.
`nsap`	Activates and displays network service access points (NSAPs).
`rtl`	Loads a static IP routing table created by the Network Directory Compiler.
`sndcmap`	Downloads a mapping table to the SNDCF driver, which enables the SNDCF-generated "called DTE" address to be mapped to the appropriate port on the WAN card.
Network, ES-IS	
`addnsap`	Adds a static entry to the ES–IS dynamic routing table.
`del`	Deletes a static entry from the ES–IS dynamic routing table.
`esq`	Displays the NSAP and SNPA address of each end system currently reachable through the CLNP dynamic routing table.
`isq`	Displays the NSAP and SNPA addresses of each intermediate system attached to the local subnetwork(s).

Continued on next page

Table 1-1. LAN Management Utilities Grouped by Layers and Protocols (cont'd)

Utility	Description
Network, ES-IS (cont'd)	
`rtgconfig`	Configures the ES–IS routing parameters.
`rtgstat`	Displays routing table statistics maintained by the ES–IS routing exchange protocol.
Data Link	
`mac`	Displays management statistics that a MAC driver maintains.
`pingllc`	Sends test packets to a remote station and verifies that the packet is returned correctly.

1.4.3. Using LT–610 With WT–325

An end system on a LAN can be installed with software that provides a simple routing service over a wide area network. The configuration for such a system requires SNDCF (subnetwork dependent convergence function).

An SNDCF STREAMS driver is provided in LT–610 for interfacing with the WT–325 or other wide–area networking product. WT–325 allows access to public and private wide area networks. Figure 1-4 illustrates the linking of the LT–610 and WT–325 stacks: the shaded boxes are IRIS OSI Transport Service modules.

SNDCF is used to convert from the connectionless-mode service that CLNP expects to the connection-oriented service provided by the X.25 packet-level protocol via the WAN stack. SNDCF links to an X.25 interface driver.

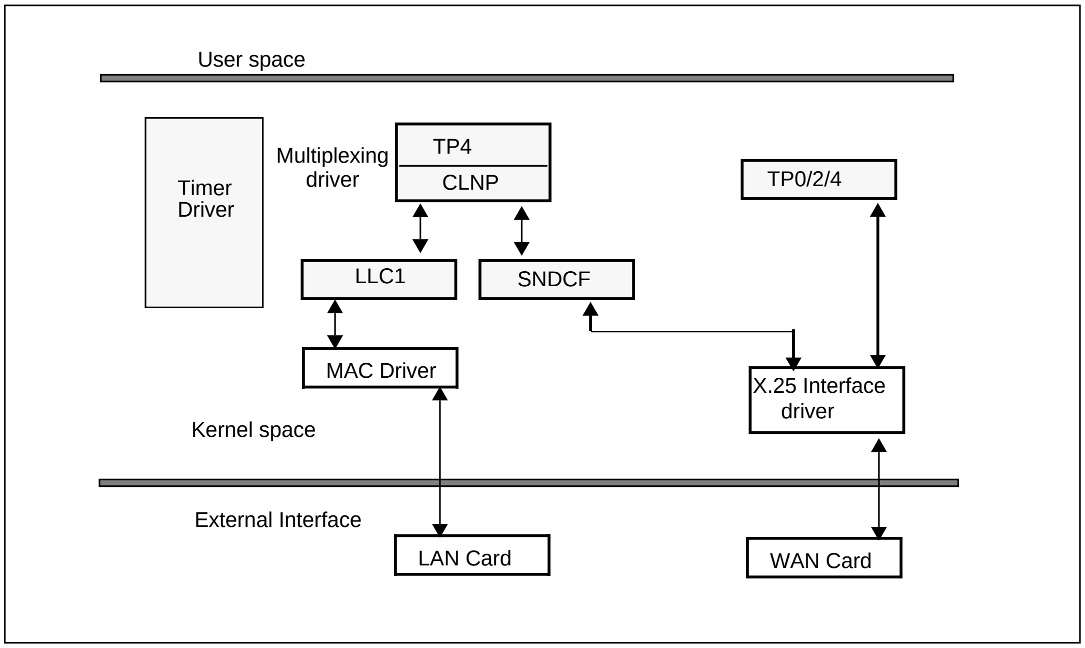


Figure 1-4. LT–610 and WT–325 Protocol Stacks

1.5. Introducing WT–325

This section introduces the components of the WT–325 WAN Transport software package and WT–325 installation and configuration utilities and files. The rest of this section introduces the following topics:

- WT–325 Transport software
- WT–325 utility programs for WAN management

1.5.1. WT–325 Transport Software

The WT–325 WAN Transport package includes the following software modules:

- **Transport 0/2/4 pseudo–driver**: implements OSI transport protocols classes 0, 2, and 4 in the STREAMS environment. Transport is accessed through the standard Transport Layer Interface (TLI), which provides the user with a function call interface to the OSI Transport Layer Services.
- **Timer driver**: provides common scheduling functions for all WT–325 drivers.

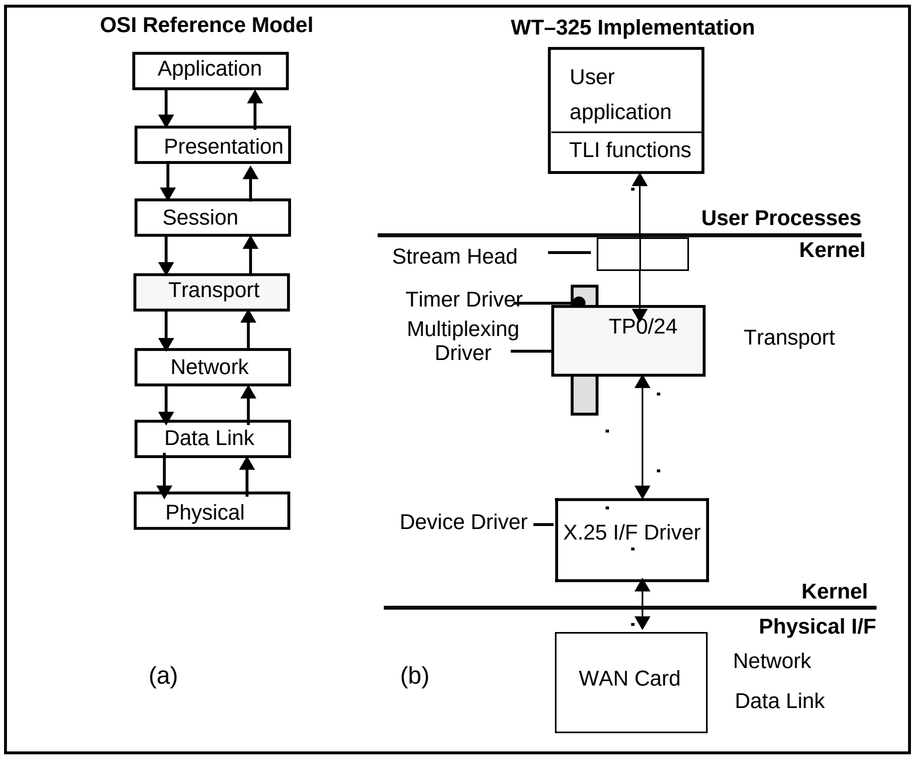


Figure 1-5. WT–325's OSI Protocol Stack

1.5.2. WT–325 Utility Programs

Table 1-2 lists the utility programs that are used to manipulate various aspects of a WAN protocol stack. Chapter 5 discusses each in detail.

Table 1-2. WAN Transport Management Utilities

Utility	Description
Transport	
`dtemap`	Downloads a mapping table to the TP0/2/4 driver, thus allowing a called NSAP to be mapped to a DTE address (required by the X.25 protocol software on the WAN card), and an associated port on the card.
`tp24stat`	Displays the management statistics that the transport layer maintains.
`tp24config`	Configures various transport layer counters, timers, and data unit formats.

1.6. Configuring the IRIS OSI Transport Service

OSI configuration utility and file (**`osid`** and `osid.cfg`) are used together to initialize the OSI stack in the STREAMS environment. These are their functions:

- **`osid`** runs as an IRIX daemon process. **`osid`** initializes the IRIS OSI STREAMS driver in use, creates the separate streams required by the Transport 0/2/4 and X.25 interface drivers, and links all the drivers together in their correct sequence.
- `/etc/osid.cfg` specifies the device names of each STREAMS driver to be combined into the STREAMS stack. `osid.cfg` contains information used by **`osid`** to configure the protocol stack. It is automatically created when you install IRIS OSI Transport Service. `osid` determines the correct devices to open as it configures the stack by reading the information in `osid.cfg`. If you change the product configuration after installing the IRIS OSI Transport Service software, you must change `osid.cfg` or errors may occur.

NOTE: IRIS OSI Transport Service software must be properly configured before any of the network management utilities can be used.

`osid` and `/etc/osid.cfg` are described in detail in Chapter 3.

1.7. Starting and Stopping IRIS OSI Transport Service

This section tells you how to start and stop your IRIS OSI LAN and WAN Transport Service.

1.7.1. Starting the IRIS OSI LAN Transport Service

1. Run the **`osid`** daemon, which automatically links STREAMS drivers for the IRIS OSI Transport Service lower layers. Enter the command:

   ```
   /usr/bin/osi/osid
   ```

2. You must now give your system a network entity address. You do this with the `net` utility provided by LT-610, which is described in Chapter 4. Addressing formats are documented in Chapter 2 of the *IRIS OSI Transport Service Network Directory Compiler Guide*, supplied with this product.

   To assign a network entity address, use the command:

   ```
   /usr/bin/osi/net <address>
   ```

3. You next assign an NSAP to your system, using the `nsap` utility described in Chapter 4. Use Chapter 2 of the *IRIS OSI Transport Service Network Directory Compiler Guide* to assist you in formatting your NSAP address. Enter the command:

   ```
   /usr/bin/osi/nsap <NSAP_address>
   ```

1.7.2. Starting the IRIS OSI WAN Transport Service

1. Ensure that the IRIS X.25 software has been downloaded to the WAN card. This may be done by invoking **`uix25`** (the IRIS X.25 user interface) or manually by running a shell script.
2. Ensure that the configuration file, `/usr/x25/x25cfg333.x`, contains appropriate entries for the number of WAN cards in use (x=2 for **one** WAN card, x=3 for second additional WAN card). For further information, please refer to the IRIS X.25 documentation, or consult the section entitled *Load X.25 Code and Configuration* in the man page for **`uix25`**.
3. Start the `osid` daemon, which automatically links STREAMS drivers for the IRIS OSI Transport Service lower layers. Enter the command:

   ```
   /usr/bin/osi/osid
   ```
4. If you are using multiple WAN ports, you may also need to download the NSAP-to-DTE mapping information required by the Transport Service software, using the **`dtemap`** utility. This utility (and the circumstances in which you need it) is described in section 5.2, on page 5-2.

1.7.3. Stopping the IRIS OSI Transport Service

1. To stop the IRIS OSI Transport Service, issue the command:

   ```
   killall -v osid
   ```

1.8. Test Utilities

Sample test programs (`sttxt` and `strxt`) can be used to demonstrate the use of the transport library and test the IRIS OSI Transport Service software (and supporting hardware) after installation and configuration. These test programs are fully described in Appendix A.

1.9. Routing Utility

The `addnsap` utility provides a means of entering routing information manually. See Chapter 4 for details.

The Network Directory Compiler command (`nd`) is sometimes needed to create routing tables that OSI network layer protocols use to make routing decisions. It is provided with IRIS OSI Transport Service for environments that do not use ES-IS, and for intermediate systems. `nd` creates static routing tables from network description files you create. See the *IRIS OSI Transport Service Network Directory Compiler Guide* or the on-line man pages for more information on using the `nd` command.

The `ndsim` command simulates the operation of the Network Directory Compiler, taking a routing table generated by `nd`, and outputting the routes (if any) to a given address.

2. SELECTING CONFIGURATION PARAMETERS

This chapter provides an introduction to the topic of selecting the appropriate configuration of your network, based on the type of connectivity you require.

Contents

2.1. Introduction

Several organizations have developed "profiles" that define standard networking configurations. These profiles were created to promote standardization of internetworking systems. Whereas OSI documents standardize the individual protocols used in open systems, these profiles standardize combinations of protocols used.

The LT–610 protocol stack can be configured to conform to the following profiles:

- US GOSIP 1.1
- UK GOSIP 3.0
- COS 1.1
- TOP 3.0
- NIST SP 500-162

The WT–325 protocol stack can be configured to conform to the following profiles:

- US GOSIP 1.1
- UK GOSIP 3.0
- NIST SP 500-162
- COS 1.1
- SPAG
- CEN/CENELEC
- TOP 3.0
- MAP 3.0
- Public X.400 services

2.2. Terminology Note

In this chapter, a distinction is made between wide area *networks* and wide area *subnetworks*. The term *Wide Area Network* refers to a complete network, consisting of one or more public or private networks, and possibly one or more Local Area Networks that can communicate with each other over the wide area network. A wide area subnetwork is a specific network or type of network that provides communications between systems over relatively long distances.

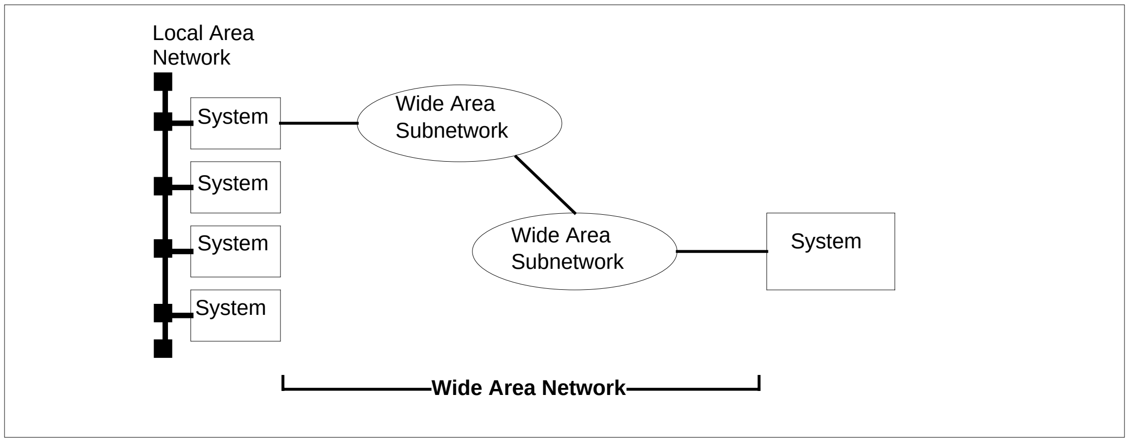


Figure 2-1. Wide Area Networks and Subnetworks

2.3. Choosing a Network Layer Protocol Combination

The most basic configuration task you must perform is choosing a network layer protocol combination appropriate for the network and application you will be using for wide area operation.

You select the network layer protocol configuration you want by setting parameters during LT–610 installation.

The criteria for selecting the correct network configuration are:

- The "profile" to which the protocol stack is designed to conform (listed in Table 2-1 on page 2-7).
- The type of network layer service the user of the network service requires.
- The type of service the wide area network provides.

This section describes these items and how they are related to LT–610 and WT–325.

2.3.1 Protocol Combinations Provided by LT–610 and WT–325

The protocol combinations used by LT–610 in conjunction with WT–325 are shown in Figure 2-2.

Figure 2-2a shows the protocol stack that LT–610 sets up. The stack provides CLNS to the network user, employing the IP protocol at the network layer, and LLC1 at the Data Link layer.

Figure 2-2b shows the protocol stack that LT–610 and WT–325 set up. This stack provides CLNS to its service user, but communicates over an X.25 public or private network (shown in Figure 2-3. Note that this stack can be used to conform to the profiles that specify CONS since it provides both CLNS and CONS to the NSU.

Figure 2-2c shows the protocol stack that WT–325 sets up. The stack provides connection–oriented Network service (CONS) to the NSU, and communicates over a public or private X.25 subnetwork (shown in Figure 2-3 on page 2-9).

NOTE: This diagram shows possible protocol combinations rather than the exact relationship between software modules.

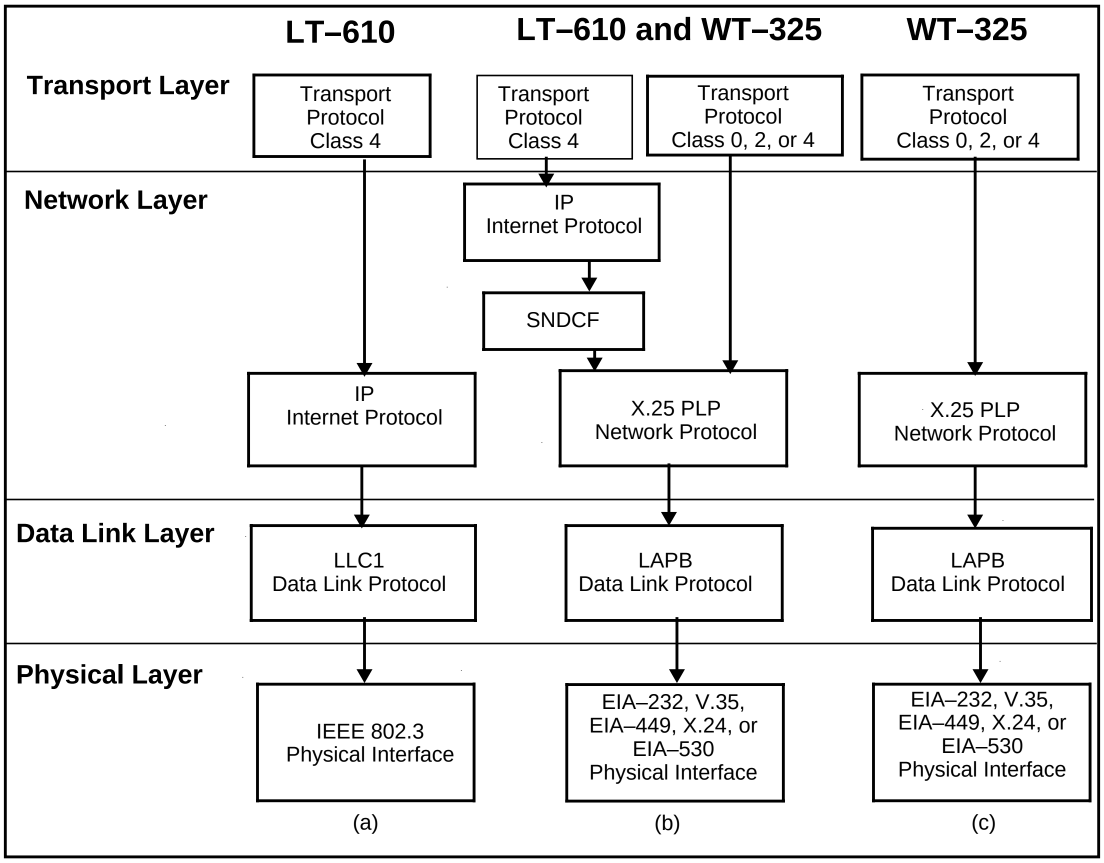

LT–610
LT–610 and WT–325
WT–325
Transport Layer
Network Layer
Data Link Layer
Physical Layer
Transport Protocol Class 4
Transport Protocol Class 4
Transport Protocol Class 0, 2, or 4
Transport Protocol Class 0, 2, or 4
IP Internet Protocol
SNDCF
IP Internet Protocol
X.25 PLP Network Protocol
X.25 PLP Network Protocol
LLC1 Data Link Protocol
LAPB Data Link Protocol
LAPB Data Link Protocol
IEEE 802.3 Physical Interface
EIA–232, V.35, EIA–449, X.24, or EIA–530 Physical Interface
EIA–232, V.35, EIA–449, X.24, or EIA–530 Physical Interface
(a)
(b)
(c)

Figure 2-2. Network Layer Protocol Combinations Provided by WT–325 and LT–610

2.3.2. Selecting the Appropriate Product Combination

To ensure compatibility with other internetworking systems, it is **strongly** recommended that you use the guidelines provided below to select the appropriate product to use.

If you are connecting one or more systems to an existing network, you must choose the product combination appropriate for the existing network. You must determine which of the network protocol combinations you should select for the applications and network you are planning to use, and select the corresponding product(s) to use. See Table 2-1.

On the other hand, if you are designing a network, you must choose an appropriate Network layer protocol combination based on the following guidelines:

- In general, protocol stacks operating in North America should provide CLNS. CLNS is required by the US GOSIP, TOP 3.0, MAP 3.0, and COS profiles for most applications (see the exception described in the next item below). In addition, many European profiles allow the optional use of CLNS. For communication over an X.25 subnetwork, use LT–610 in conjunction with WT–325.
- Both North American and European profiles specify CONS for use with public X.400 applications. The correct product to select for use with these applications is WT–325.
- UK GOSIP specifies that the network layer must provide CONS. You would normally use WT–325 for protocol stacks that conform to this profile.
- When dedicated lines are used for wide area communications, you would normally use the same protocol for network service and subnetwork access. Choose WT–325 for communication over a private X.25 subnetwork.
- CONS should be used with non–OSI applications that access the network protocol directly (rather than through transport). Use WT–325 with these applications.

Table 2-1. Criteria for Selecting Product(s)

Profile	Media Type	Network Service	Products
US GOSIP, COS MAP TOP	X.25 subnet	CLNS	LT–610 & WT–325
Public X.400, SPAG T/311, T/23, Y/11, UK GOSIP CEN/CENELEC ENV 41104, 41901	X.25 subnet	CONS	WT–325
UK GOSIP SPAG T/21	Private X.25	CONS	WT–325

2.4. Types of Wide Area Networks

There are three major characteristics that differentiate wide area networks:

- Network layer protocol
- Access type
- Network media

Network layer protocol. OSI systems use one of two network protocols to communicate over a wide area subnetwork. One protocol is called X.25 Packet Level Protocol (X.25 PLP). X.25 PLP is a connection–oriented protocol. For two systems to communicate, their X.25 PLP entities must establish a connection before the transfer of data can occur. The other protocol is called Internetwork Protocol (IP). IP is a connectionless protocol; when a system has data to send to another system, it simply sends a *datagram* addressed to the other system. A discussion of the differences between connectionless and connection–oriented network protocols is beyond the scope of this manual. For the purpose at hand, it is sufficient to note only that in order for two systems to communicate over a subnetwork, both must use the same protocol; in addition, the protocol must be the same one used by the intermediate systems within the subnetwork.

WT–325 provides only the connection–oriented network service. LT–610 provides a connectionless network service.

NOTE: Some literature calls Internetwork Protocol, Connectionless Network Protocol, or CLNP.

Access type. A subnetwork can be public or private. A publicly accessible network, called a *public data network*, is accessible to any organization that has the proper equipment and agrees to pay access costs charged by the organization that operates the network. If a network does not operate under these terms, it is a private data network. Such networks limit their accessibility to a specific organization or group of organizations. In general, public networks use X.25 PLP. Private networks may use IP or X.25 PLP depending on the profile on which the network is based.

Network media. The media that transfers data between end systems may be a single private line or one or more intermediate systems. The network protocol that may be used over the private line can be either X.25 PLP or IP.

Public data networks that use X.25 PLP consist of a series of intermediate systems called *switches* to route data between systems. Private networks may also be configured using one or more X.25 PLP switches.

A private network can consist of a series of IP *routers*. Like X.25 switches, IP routers are intermediate systems that route data between end systems. (Strictly speaking, X.25 switches are actually routers. However, the term “switches” is more commonly used to describe X.25 intermediate systems.)

2.5. Types of Network Service

Each of the profiles specifies the type of network service a network service user (NSU) of that profile requires. There are two types of OSI network service: connection–oriented network service (CONS) and CONS is implemented by X.25 PLP, whereas CLNS is implemented by IP.

Figure 2-3(a) and Figure 2-3(b) show networks in which the protocol used to access the network is the same one that provides service to the NSU. Figure 2-3(c) shows a network in which IP provides CLNS to the NSU, but uses X.25 PLP to access the wide area subnetwork. The SNDCF (subnetwork dependent convergence functions) protocol shown in the figure maps service requests made by CLNS to X.25 PLP operations.

Note that all of the CLNS NSUs shown in Figure 2-3(a) and Figure 2-3(b) can interoperate. NSUs that use CONS cannot interoperate directly with those that use CLNS.

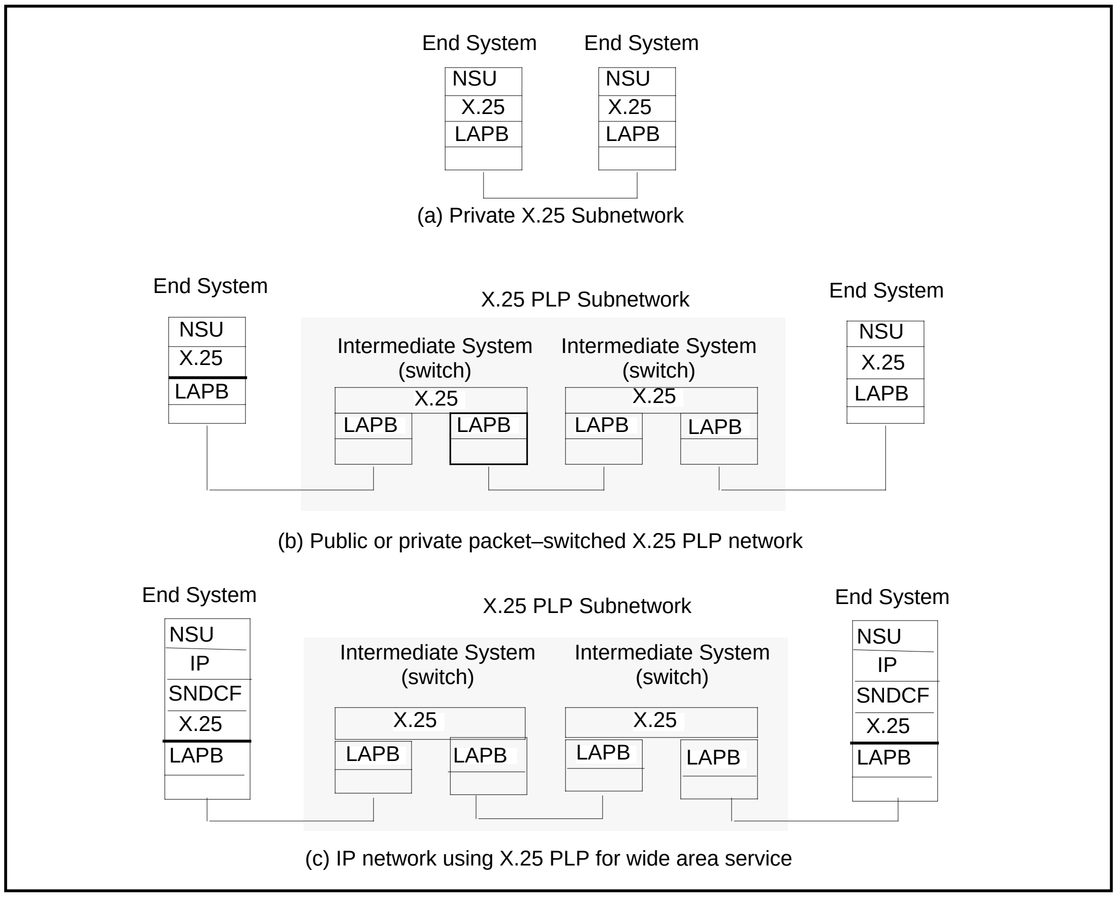

End System
End System
NSU
X.25
LAPB
NSU
X.25
LAPB
(a) Private X.25 Subnetwork
End System
X.25 PLP Subnetwork
End System
NSU
X.25
LAPB
Intermediate System (switch)
Intermediate System (switch)
X.25
LAPB
LAPB
X.25
LAPB
LAPB
NSU
X.25
LAPB
(b) Public or private packet–switched X.25 PLP network
End System
X.25 PLP Subnetwork
End System
NSU
IP
SNDCF
X.25
LAPB
Intermediate System (switch)
Intermediate System (switch)
X.25
LAPB
LAPB
X.25
LAPB
LAPB
NSU
IP
SNDCF
X.25
LAPB
(c) IP network using X.25 PLP for wide area service

Figure 2-3. Network Service and Access Protocol Combinations

3. STACK CONFIGURATION

This chapter tells you how to configure your IRIS OSI Transport Service stack to reflect your choice of network service. The necessary steps are:

•editing of the `osid.cfg` file to meet your requirements

•running **`osid`**

Contents

3.1. Introduction

The `osid` STREAMS configuration utility constructs the STREAMS protocol stack from drivers provided with LT–610, WT–325, and other STREAMS products. This chapter describes how to modify the contents of the `osid.cfg` file to reflect the desired protocol stack configuration **before** running the **`osid`** STREAMS configuration utility.

Sample scenarioes are used to illustrate the ways in which the `osid.cfg` file can be used to configure STREAMS. The Chapter then looks at the issue of setting timer valuesa final selection examines the use of the **`osid`** configuration utility.

3.2. The STREAMS Configuration File: `osid.cfg`

The `osid.cfg` STREAMS configuration file specifies which STREAMS drivers have been installed and which ones are to be used to build the protocol stack.

The file consists of a series of entries. Each entry corresponds to an installed STREAMS driver. An entry consists of a symbol that represents the driver, followed by one or more parameters applicable to the driver:

```
Symbol param1=value1 [param2=value2] ...
```

Comments may be included in the configuration file. A comment line begins with a # character.

The purpose of each entry is to provide a permanent name to each STREAMS driver, and to associate this name with the name of a STREAMS device (which can be changed) accessible to a user program. The configuration utility can thus find the UNIX device name of each STREAMS driver by searching for its symbol in the `osid.cfg` file.

There are two types of entries in `osid.cfg`: *specific entries* and *generic entries*. The configuration utility uses specific entries to determine which STREAMS drivers must be linked together. The configuration utility is designed to automatically build the correct protocol stack based on the specific entries in the `osid.cfg` file.

Generic entries can be read by networking applications that use OSI transport products. These entries allow a system administrator to select which protocol stack the application accesses this capability is useful if two or more similar protocol stacks are installed. In addition, the configuration utility reads some generic entries in cases where specific entries do not provide enough information to configure a protocol stack.

3.2.1. Specific Entries in `osid.cfg`

The specific entries for the STREAMS drivers are listed below. Note that each entry has a device name parameter. Other parameters associated with the entry are also explained.

`TIMER device=<`*`timer_device`*`>`

The timer driver. The default timer device name is `/dev/tmr`. It is used by **both** LT–610 and WT–325.

`SYSTEM pollbug=TRUE`

This value is used by **both** LT–610 and WT–325.

`TP_610 device=<`*`tp4_device`*`>`

The entry for the driver that incorporates the connection–oriented Transport Protocol Class 4 (TP4) and connectionless Internetwork Protocol (IP), used by LT–610. The default name for this device is `/dev/cots`.

`DLPI_610 device=<`*`dlp_device`*`>`

The entry for the driver that supports LLC1 service primitives, which is used by LT–610. The default name for this device is `/dev/dlr`.

`SNDCF_610 device=<`*`SNDCF_device`*`>`

The entry for the Subnetwork D

ependent convergence functions (SNDCF) driver. SNDCF, in conjunction with WT–325, provides an interface to a wide area public or private data network. The default name for this device is `/dev/sndc`.

`PORT`*`n`*` device=<`*`MAC_device`*`> [unit=`*`u`*`] [esgaddr[=`*`a1`*`]] [isgaddr[=`*`a2`*`]]`

`PORT`*`n`* is the entry for a MAC driver; ***`n`*** is a non–negative decimal integer that identifies a particular MAC driver. (It also specifies a logical port for the purposes of creating a routing table using the Network Directory Compiler.) In the first instance of **`PORT`** in the configuration file, ***`n`*** must be **`0`**. In each subsequent instance, ***`n`*** must be incremented by 1. Default `PORT0` device = `/dev/snif.`

`MAC_device` is the device name of the MAC driver you are using. The `/dev/snif` driver is used by LT–610. Other drivers may be used, provided that they supply the required DLPI interface.

`unit` is the unit number of the physical device associated with the MAC driver.

`a1` is the end system multicast address of the device; and ***`a2`*** is the intermediate system multicast address. If you include **`esgaddr`** or **`isgaddr`** without specifying the address, the value defaults to the standard multicast address.

`TP_325 device=<`*`TP024_device`*`>`

The entry for the transport classes 0/2/4 STREAMS interface driver (used by WT–325). The default name for this device is `/dev/cox`.

`MX25_`*`n`*` device=<`*`X25_device`*`> loc_dte=<`*`local_DTE_address`*`> nvcs=`*`number`*

The entry for the X.25 interface driver (used by WT–325). The default name for this device is `/dev/sx25`.

At its lower interface, this driver interfaces to the WAN card(s): a maximum of 2 cards is supported, and each card has 2 ports.

In the **`MX25_`*`n`*** entry, ***`n`*** specifies a particular combination of CPU (i.e., card) and port, described in Table 3-1 below. CPU-ID 2 refers to a single WAN card, and CPU-ID 3 refers to an additional WAN card. Note that, unlike the **`PORT`*`n`*** entries, the **`MX25_`*`n`*** entries may appear in any order.

Table 3-1. Specifying Port and Card Combinations

Value of *n*	Port and Card Combination
0	CPU-ID 2, Port 0
1	CPU-ID 2, Port 1
2	CPU-ID 3, Port 0
3	CPU-ID 3, Port 1

The **`loc_dte`** parameter specifies the local DTE address, associated with this particular port / card combination. The length of the DTE address can range from 1 to 15 BCD (Binary Coded Decimal) digits.

The **`nvcs`** parameter specifies the number of X.25 virtual circuits that TP 0/2/4 or SNDCF can use, excluding those used for handling incoming calls.

3.2.2. Generic Entries in `osid.cfg`

User programs use the following generic entries in `osid.cfg` to determine which devices to open:

- **`TP024`** (Default is `/dev/cox`)
- **`TP4`** (Default is `/dev/cots`)
- **`SNDCF`** (Default is `/dev/sndc`)
- **`LLC1`** (Default is `/dev/dlr`)
- **`PORT0`** (Default is `/dev/snif`)

Generic entries are read by user applications to determine which stack to use, when two or more similar protocol stacks are present. All generic entries require a single parameter: the device name of the STREAMS driver with which the entry is associated.

3.3. Example Configuration File Scenarios

The configuration utility is designed so that it automatically builds the correct protocol stack based on the specific and generic entries that the `osid.cfg` file contains. The requirements of different network environments are shown in the examples below.

3.3.1. Configuring a LAN End System

The following is an example of an `osid.cfg` file designed to meet LAN end system configuration.

```
# Generic module names
TP4             device=/dev/cots           # TP_610
LLC1            device=/dev/dlr            # DLPI_610

# MAC driver port 0 [ SGI Ethernet Card ]
PORT0     device=/dev/snif unit=0 esgaddr

# Mandatory lower layer support timer [ TIMER ]
TIMER   device=/dev/tmr

# LT–610 Class 4 Transport over CLNS [ TP_610 ]
TP_610 device=/dev/cots

# LT–610 LAN support [ DLPI_610 ]
DLPI_610         device=/dev/dlp
```

Comments

This end-system configuration file contains definitions for Transport Protocol Class 4 and Data Link layer drivers. The System pollbug variable is set to true (as in all of the scenarios given in this section). Just one port (**PORT0**) is specified, and this port uses end-system group addressing (**esgaddr**).

3.3.2 Configuring a LAN Intermediate System

The following is an example of an `osid.cfg` file designed to meet LAN intermediate system configuration requirements.

```
# Generic module names
TP4             device=/dev/cots
LLC1            device=/dev/dlr

# Mandatory lower layer support timer [ TIMER ]
TIMER   device=/dev/tmr

# LT–610 Class 4 Transport over CLNS [ TP_610 ]
TP_610  device=/dev/cots

# LT–610 LAN support ( required if LAN supported )
# [ DLPI_610 ]
DLPI_610         device=/dev/dlr

# MAC driver port 0 [ SGI Ethernet Card ]
PORT0     device=/dev/snif  unit=0  isgaddr

# MAC driver port 1 [ SGI Ethernet Card ]
PORT1     device=/dev/snif  unit=1  isgaddr
```

Comments

This intermediate system configuration file contains definitions for Transport Protocol Class 4 and Data Link Layer drivers. Note however, that two ports (**PORT0** and **PORT**1) are in use, and that intermediate system group addressing (**isgaddr**) is employed.

3.3.3 Configuring a LAN/WAN End System to Intermediate System

The following is an example of an `osid.cfg` file designed to meet end system to intermediate system configuration.

```
# Generic module names
TP4             device=/dev/cots                # TP_610
SNDCF           device=/dev/sndc               # SNDCF_610
LLC1            device=/dev/dlr                 # DLPI_610
TP024           device=/dev/cox                 # TP_325

# Mandatory lower layer support timer [ TIMER ]
TIMER   device=/dev/tmr

# Class 0/2/4 transport service provider [ TP_325 driver ]
TP_325 device=/dev/cox

# LT-610 Class 4 Transport over CLNS [ TP_610 ]
TP_610 device=/dev/cots

# LT-610 LAN support ( required if LAN supported )
# [ DLPI_610 ]
DLPI_610        device=/dev/dlr

# MAC driver port 0 [ SGI Ethernet Card ]
PORT0     device=/dev/snif unit=0 esgaddr

# LT-610 WAN support ( required if CLNS over X.25
# supported ) [ SNDCF_610 ]
SNDCF_610       device=/dev/sndc

# X.25 Interface Driver Entries
MX25_0 device=/dev/sx25 loc_dte=12345678901111 nvcs=4
```

Comments

This end system to intermediate system configuration file contains definitions for Transport Protocol Class 4, SNDCF, Transport Class 0/2/4, and Data Link Layer drivers. One port (**PORT0**) is in use, and end-system group addressing (**esgaddr**) is employed. The number of virtual circuits (**nvcs**) is set to 4.

The last entry references the X.25 interface driver: in this case, Port 0 on CPU-ID 2.

3.3.4 Configuring a WAN System

The following is an example of an `osid.cfg` file designed to meet WAN requirements.

```
# Generic module names
TP024           device=/dev/cox                  # TP_325

# Mandatory lower layer support timer [ TIMER ]
TIMER   device=/dev/tmr

# Class 0/2/4 transport service provider [ TP_325 driver ]
TP_325 device=/dev/cox

# X.25 Interface Driver Entries
MX25_1 device=/dev/sx25 loc_dte=12345678901111 nvcs=4
```

Comments

This WAN configuration file contains definitions for Transport Protocol Class 0/2/4 and X.25 interface drivers. **`MX25_1`** specifies CPU-ID 2, Port 1 (in other words, a single WAN card, or the first WAN card if two are installed). The number of virtual circuits (**`nvcs`**) is set to 4.

3.4. Transport Timer Recommendations

This section discusses timer settings for the OSI transport protocols TP0, TP2, and TP4. It is meant to give you insight into how the default value for a timed event is derived. Various factors are taken into consideration and weighed to arrive at the default timer setting for general use. See the **`tp24config`** output example in section 5.3 on page 5-4, for a listing of the transport timers and associated defaults. The timed event used in the following example is the retransmission of packets.

Assume that unusual network conditions are all transient in nature and there is 100 percent line utilization. Should this assumption not be correct, scale the results appropriately, making the numbers bigger to account for poorer performance. LAPB and X.25 window sizes less than 7 will cause significant performance decreases.

If Transport attempts to send a packet, the worst case exists if all the other transport connections have already sent their full amount of data and that this packet is queued behind all of them. The maximum queued bytes outstanding is expressed in this equation:

Max Q’d bytes outstanding at any moment = size of each TIDU * Window Size * X Connections

TP0 and TP2 do not attempt retransmissions, but TP4 does. To prevent TP4 from retrying too soon, you should set the Class 4 parameters to their default values. The default retransmission timer value is derived from the throughput that is computed for each TP4 connection using this algorithm:

•Each TP4 connection would get a throughput of about:

$$\left(\frac{\text{8 bits per byte * Max Q'd packets}}{\text{Line throughput in bits per second}}\right) \text{seconds per packet}$$

•Inputs into the above expression are obtained from these sources:

- For X connections, get the **`nvcs`** value associated with the **`MX25_n`** entry in `/etc/osid.cfg`.
- For TIDU size, use 512 bytes, which is the fixed value for the WT-325 implementation.
- For window size, use the transport default window size of 4, which is adjustable via `tp24config`.
- For line throughput, use the line speed of 9600 bits per second.
- So 8 connections at 9600 baud:

$$\frac{\text{8 bits/byte * (512 bytes/packet * 4 windows * 8 connections)}}{\text{9600 bits/sec}}$$

yields about 15 seconds per packet.

Thus, the TP4 retransmission timer **must** be greater than 15 seconds. If not, X.25 PLP or LAPB will become clogged with TP4 retries that have not even had time to transmit.

Now back to the packet under discussion. If the packet is sent, it could face the same delay on reception, so the timer should wait 30 seconds. The recommended default value, then, for the retransmission timer is 30 seconds.

3.5. The STREAMS Configuration Utility: `osid`

The **`osid`** STREAMS configuration utility constructs the STREAMS protocol stack from LT–610 and WT–325 STREAMS drivers. **`osid`** can also build drivers provided with other IRIX STREAMS products into the STREAMS protocol stack. **`osid`** determines which drivers to include in the protocol stack by reading the `osid.cfg` configuration file.

3.5.1. Tasks Performed by the STREAMS Configuration Utility

The sequence of steps that the **`osid`** utility performs is described below.

1. The **`osid`** utility forks a daemon process, then terminates. The daemon process continues running to maintain the protocol stack.
2. **`osid`** opens the timer driver.
3. If the TP0/2/4 driver is available (part of the WT–325 product), **`osid`** builds the connection–oriented network service (CONS) stack by opening the TP0/2/4 driver and linking a number of X.25 Streams underneath it.
4. If the TP4/IP driver (part of the LT–610 product) is present, **`osid`** builds the connectionless stack. If the LLC1 driver is present, it links one or more MAC drivers under the LLC1 driver, then links the LLC1 Stream under the TP4/IP driver.
5. If the TP4/IP driver is present, and the SNDCF driver is present in addition to or instead of the LLC1 driver, **`osid`** opens SNDCF, links a number of X.25 PLP Streams under it, and links the SNDCF Stream under the TP4/IP Stream.

4. LT–610 UTILITIES

This chapter discusses the network management utilities provided with the LT–610 product. These utilities set and display the values of configuration parameters and manage routing tables. The utilities are presented in alphabetical order.

NOTE: All LT–610 Network Management utilities require superuser privileges.

Contents

4.1. Add a Static Entry to the IP Routing Table: addnsap

Purpose

The `addnsap` utility allows you to add the addresses of remote systems to the local CLNP routing table. When ES–IS routing exchange protocol is being used, the user does not normally have to be aware of this table; it is modified dynamically as remote systems go on–and off–line. However, this protocol may not always be able to provide the necessary routing information for all reachable systems (for example, systems that do not use the ES–IS protocol). The `addnsap` utility provides a means of entering the routing information manually.

Command Line

`addnsap` ***`NSAP_Address SNPA_Address [port]`***

Arguments

`NSAP_Address`

NSAP address of the remote system. Only hexadecimal digits are accepted. Do not punctuate the NSAP address with spaces, periods, or other characters.

A description of NSAP address formats can be found in the *IRIS OSI Transport Service Network Directory Compiler Guide*.

`SNPA_Address`

SNPA address of the remote system (if the system is on the same subnetwork) or of the first intermediate system (if the system is on a different subnetwork than the local system) in the path to the remote system. The SNPA address consists of 7 bytes. The first 6 bytes identify the factory-installed MAC address; the seventh specifies the LSAP address (always FE). Only hexadecimal digits are accepted. Do not enter spaces or other punctuation.

If you are adding a system reachable through a wide area network (using a product such as the WT–325 WAN Transport package), the SNPA is in X.121 format (up to 15 decimal digits).

`port`

An optional argument that specifies the MAC driver that is attached to the subnetwork over which the remote system is accessed. The port number of a MAC driver is specified in the STREAMS configuration file.

If you are adding a system reachable over a wide area network, specify the port number of the SNDCF driver. This number is one higher than the port number of the last MAC driver.

If you do not specify a port, the default is 0.

Notes

This utility works only with systems using ES–IS.

Command Line Example

```
addnsap 490004000600010207010l2b5501 0207010061B1FE
```

Error Messages

```
usage: addnsap dst snpa [port]
```

Cause: You entered the command line arguments incorrectly.

Action: Reenter the command line making sure you enter at least the correct destination NSAP address and SNPA.

```
add: invalid NSAP–address length
```

Cause: The NSAP address you entered was too long or contained an odd number of characters.

Action: Check the NSAP address and reenter the corrected command line. The address must contain an even number of hexadecimal characters (since a byte consists of 2 digits), and its length must be 40 characters or less.

```
add: invalid SNPA–address length
```

Cause: The SNPA address contained the wrong number of characters.

Action: Check the SNPA address and reenter the corrected command line.

`add: invalid hexadecimal character` *`c`* `in address`

Cause: A character that was not a valid hexadecimal digit was specified in either the NSAP or SNPA address.

Action: Check the NSAP or SNPA address, and reenter the command line. These addresses must contain only the characters 0 – 9 and a – f (or A – F).

4.2. Delete a Routing Table Entry: `del`

Purpose

Delete one entry in the IP routing table.

Command Line

`del` ***`NSAP_Address`***

Argument

`NSAP_Address`

NSAP address of the remote system whose path is to be deleted.

Notes

This utility can be used only on systems using ES–IS.

This utility does not check whether the routing table entry you wish to delete was added using the `addnsap` utility or was added by the ES–IS routing exchange protocol. If ES–IS added it, you can delete it temporarily, but it will likely reappear after several seconds.

A description of NSAP address formats may be found in the *IRIS OSI Transport Service Network Directory Compiler Guide*.

Command Line Example

```
del 49000400060001020701012b5501
```

Error Messages

`usage: del nsap–address`

Cause: You entered the command line arguments incorrectly.

Action: Reenter the command line making sure you enter one valid NSAP address.

`del: invalid NSAP–address length`

Cause: The NSAP address you entered was too long or it contained an odd number of characters.

Action: Check the NSAP address and reenter the corrected command line. The address must contain an even number of hexadecimal characters (since a byte consists of two digits), and its length must be 40 characters or less.

`del: invalid hexadecimal character` *`c`* `in address`

Cause: A character that was not a valid hexadecimal digit was specified in the NSAP.

Action: Check the NSAP address, and reenter the command line. NSAP addresses must contain only the characters 0 – 9 and a – f (or A – F).

4.3. View All Reachable End System Addresses: esq

Purpose

Display the NSAP and SNPA addresses of all end systems that have entries in the ES–IS IP routing table.

Command Line

```
esq
```

Notes

In addition to the NSAP and SNPA addresses, this utility also displays the “hold” time for each remote system. This value indicates in how many seconds the entry will be removed if the local system does not receive a hello packet from the remote system. If the hold time is listed as 0, the entry was added statically and will never be automatically deleted.

If this command is executed from an end system, it is likely to show only end systems that were entered statically with the `addnsap` utility, those that are being accessed currently, or those that have been accessed more recently than the hold period.

If this command is executed from an intermediate system, it will show static entries and end systems that have recently transmitted “Hello” Protocol Data Units (PDUs).

This command works only on systems that use ES–IS.

Output Example

```
               NSAP address    SNPA address   Hold
4900040006000102070100061B101 0207010061B1FE    44
```

Error Message

```
esq: no End System table in this kernel
```

Cause: LT–610 software is not installed.

Action: Reinstall the LT–610 software.

4.4. Configure IP Parameters: `ipconfig`

Purpose

Sets values of IP communications parameters and displays current parameter values.

Command Line

`ipconfig` ***`[argument value] [argument value] ...`***

Arguments

`–lifetime` ***`units`***

Specifies the amount of time (in units of 500 milliseconds) that other systems will consider valid an IPDU sent from this system. If you do not specify a value for this parameter, it defaults to 20 (10 seconds).

`–checksum` ***`flag`***

If **`flag`** is 0, the IP checksum mechanism is disabled. Any other value enables it. If you do not specify a value for this parameter, it defaults to 0 (disabled).

`–report` ***`flag`***

If **`flag`** is 0, the error report request bit in outgoing IPDUs is not set; otherwise, it is set. This bit specifies that if an error is detected in the IPDU, the entity that detects the error may send an Error Report (ER) PDU. If you do not specify a value for this parameter, it defaults to 0 (disabled).

`–segment` ***`flag`***

If **`flag`** is 0, outgoing IPDUs cannot be segmented; otherwise, they can be segmented. If you don't include this argument, its value defaults to 1 (enabled).

Disabling segmentation is **not** recommended, and is not necessary in the normal course of network operation.

`–reasmbly` ***`units`***

Specifies the maximum amount of time (in units of 500 milliseconds) in which all segments of a IPDU must be received. If the lifetime parameter within the IPDU is smaller, it takes precedence over the value specified in this argument. If you don't include the **`reasmbly`** argument, its value defaults to 20 (10 seconds).

`–help`

Displays a list of command line options.

Command Line Example

```
ipconfig –lifetime 10 –reasmbly 50
```

Output Example

```
PDU Lifetime (units of 500 milliseconds):       10
Network layer header checksum usage:            0
Error reporting enable (on outgoing PDUs):     0
Segmentation permitted (on outgoing PDUs):     1
Reassembly timer upper bound (units of 500 ms): 50
```

Error Messages

`could not open` *`driver`*

Cause: The TP4/IP driver is not available.

Action: The most likely cause of this error is that the STREAMS stack has not been initialized. Invoke the `osid` configuration program.

`ipconfig: unknown parameter`

Cause: `ipconfig` did not recognize one or more of the arguments you entered on the command line.

Action: Check the list of argument names and reenter the command line.

`ipconfig: cannot set` *`name`* `parameter`

Cause: A value you specified was rejected by the TP4/IP module.

Action: Use a different value.

4.5. Read IP Network Management Statistics: `ipstat`

Purpose

Display the read–only statistics concerning network operation that IP keeps.

Command Line

`ipstat`

Output Description

`Number of PDU’s received`

Number of correctly formatted IPDUs this system has received.

`Number of PDU’s transmitted`

Number of IPDUs this system has transmitted.

`Number of segmented PDU’s received`

Number of IPDUs this system has received as a series of segmented data units.

`Number of segmented PDU’s transmitted`

Number of IPDUs this system has segmented and successfully transmitted.

`Number of bytes of user data received`

Number of bytes of user data contained in Data Transfer (DT) IPDUs this system has received.

`Number of bytes of user data transmitted`

Number of bytes of user data contained in Data Transfer (DT) IPDUs this system has transmitted.

`Number of PDU fragments received`

Number of incomplete (segmented) IPDUs this system has received.

`Number of PDU fragments transmitted`

Number of incomplete (segmented) IPDUs this system has transmitted.

`Number of PDU fragments discarded`

Number of incomplete (segmented) IPDUs this system has discarded.

`Number of ER PDU’s received`

Number of Error Report (ER) IPDUs this system has received.

`Number of ER PDU’s transmitted`

Number of Error Report (ER) IPDUs this system has transmitted.

`Number of discarded PDU’s unknown reasons (bad header)`

Number of times this system has received an IPDU that contained errors in the IP header and therefore could not be parsed.

`Number of discarded PDU’s caused by bad checksum`

Number of times this system has received an IPDU that contained a checksum that indicated the IPDU was corrupt.

`Number of discarded PDU’s caused by unknown NSAP address`

Number of times this system has discarded an IPDU because it contained an unknown NSAP. In end systems, this counter is incremented when the NSAP address that the IPDU contains does not match any NSAP address specified for this system. In intermediate systems, this counter is incremented when the system’s dynamic or static routing table does not contain a path to the end system whose NSAP address is contained in the IPDU.

`Number of discarded PDU’s caused by lifetime expiration`

Number of times this system has discarded an IPDU because it could not be delivered to an end system within the specified lifetime of the IPDU. IPDU lifetime expiration occurs only in intermediate systems.

`Number of discarded PDU’s caused by invalid options`

Number of times this system has discarded an IPDU because it contained an invalid type 2 option, or any syntax error in the options portion of the header.

`Number of discarded PDU’s caused by reassembly timeout`

Number of times this system has discarded a segmented IPDU because all segments of the IPDU were not received before the reassembly timer expired.

`Number of discarded PDU’s caused by lack of reassembly resources`

Number of times this system has discarded a segmented IPDU because the system did not have sufficient receive buffers or reassembly data structures available.

`Number of discarded PDU’s caused by segmentation not permitted`

Number of times this system has discarded an IPDU because the IPDU was segmented, but segmentation had been disabled.

`Number of discarded PDU’s caused by inactive subset used`

Number of times this system has received an IPDU that was built under the Inactive Network Layer Protocol (INLP) when the system’s ability to receive an INLP IPDU was disabled.

`Number of inactive subset PDU’s transmitted`

Number of IPDUs built under INLP transmitted.

`Number of inactive subset PDU’s received`

Number of IPDUs built under INLP received.

Error Message

`could not open` *`device`*

Cause: The TP4/IP module could not be opened, probably because the STREAMS stack is not initialized, or LT–610 has not been installed.

Action: Invoke the **`osid`** stack configuration utility, or install the LT–610 software.

4.6. Display All Reachable Intermediate System Addresses: `isq`

Purpose

Displays the network entity title and SNPA address of each intermediate system that has an entry in the IP routing table.

Command Line

`isq`

Notes

In addition to the network entity title and SNPA addresses, this utility also displays the "hold" time for each intermediate system. This value indicates in how many seconds the entry will be removed if the local system does not receive a hello packet from the intermediate system.

The maximum number of intermediate system entries the ES–IS IP routing table can hold is 8.

This command works only on systems that use ES–IS.

Output Example

```
       Network entity title    SNPA address Hold
490004000600010207010030A400 0207010030A4FE    61
4900040006000100AA0002305900 00AA00023059FE    29
490004000600010207010061B100 0207010061B1FE    58
```

Error Message

```
isq: no Intermediate System table in this kernel
```

Cause: LT–610 software is not installed.

Action: Install the LT–610 software.

4.7. Read MAC Entity Management Statistics: `mac`

Purpose

Reads the various statistics a MAC driver may keep.

Command Line

`mac` ***`[port]`***

Argument

`port`

An optional argument that specifies which MAC driver's management statistics you wish to view. The port number is determined by the MAC driver's unit number assigned in the STREAMS configuration file.

If no port is specified, the default is 0.

Output Description

`Number of successful transmissions`

Number of MAC frames the system has successfully transmitted.

`Number of successful frames received`

Number of MAC frames the system has successfully received.

`Number of broadcast frames received`

Number of MAC broadcast frames the systems has successfully received.

`Number of broadcast frames transmitted`

Number of MAC broadcast frames the system has successfully transmitted.

`Number of multicast frames received`

Number of MAC multicast frames the system has successfully received.

`Number of multicast frames transmitted`

Number of MAC multicast frames the system has successfully transmitted.

`Number of multicast frames rejected`

Number of MAC frames the system has rejected for any reason.

`Number of frames dropped due to lack of a STREAMS buffer`

Number of MAC frames the system has discarded because no STREAMS buffer could be allocated.

Error Messages

`PORT`*`n`*` not configured`

Cause: The port you specified does not exist.

Action: Check the STREAMS configuration file (`osid.cfg`) to determine the ports that are available.

`could not open `*`device`*

Cause: The name of the device whose port number you specified could not be opened. Either the configuration file is incorrect, or the MAC driver was installed incorrectly or is not installed in the IRIX kernel.

Action: View the contents of the `osid.cfg` file to make sure that the device name for the MAC driver is correct. If it is, you will probably have to reinstall the MAC driver into the IRIX kernel. Consult the IRIS software installation guide.

`mac: PORT`*`n`*` unit=`*`u`*

Cause: The unit number associated with the port you specified could not be activated. Either the LAN card is not installed, or it is faulty.

Action: Check the LAN card to make sure it is fully operational.

4.8. Set a Local Network Entity Title: `net`

Purpose

Set the local IP network entity title and/or read back the current value.

Command Line

```
net [title]
```

Argument

`title`

The network entity title.

Command Line Example

```
net 49000400060001020701012b5500
```

Notes

The network entity title defaults to the value specified when the LT–610 package was installed. Use this utility to change the default value.

Error Messages

```
usage: net [local–Network–Entity–Title]
```

Cause: The command line was entered incorrectly.

Action: Reenter the command line with a single network entity title specified.

```
net: no network layer in this kernel
```

Cause: The TP4/IP driver is not present.

Action: Install LT–610.

```
net: invalid address length n
```

Cause: The Network entity title you entered was too long or it contained an odd number of characters.

Action: Check the Network entity title and reenter the corrected command line. The address must contain an even number of hexadecimal characters (since a byte consists of two digits), and its length must be 40 characters or less.

```
net: invalid hexadecimal character c in address
```

Cause: The Network entity title you entered contained a character that was not a valid hexadecimal digit.

Action: Check the Network entity title, and reenter the command line. Network entity titles must contain only the characters 0 – 9 and a – f (or A – F).

4.9. Specify and/or Display NSAP Addresses: nsap

Purpose

Specifies one or more local NSAP addresses and displays the NSAP addresses that are already in use.

Command Line

nsap ***[NSAP_Address] [NSAP_Address] ...***

Arguments

If no arguments are specified, the currently activated NSAP addresses are displayed. If one or more arguments are specified, each is taken to be an NSAP address to be activated.

Notes

The maximum number of local NSAP addresses that can be used is 8.

Default local NSAP addresses are specified when LT–610 software is installed. Use this utility to specify additional NSAP addresses.

A description of NSAP address formats may be found in the *IRIS OSI Transport Service Network Directory Compiler Guide*.

Command Line Example

```
nsap 4900040006000100000C0767E1001
```

Output Example

```
NSU                     NSAP address
  1      49000400060001020701012b5501
  2      4900040006000100000C0767E1001
```

Error Messages

```
nsap: no network layer in this kernel
```

Cause: The TP4/IP driver is not present.

Action: Install LT–610.

nsap: invalid address length *n*

Cause: An NSAP address you entered was too long or it contained an odd number of characters.

Action: Check the NSAP address and reenter the corrected command line. The address must contain an even number of hexadecimal characters (since a byte consists of 2 digits), and its length must be 40 characters or less.

`del: invalid hexadecimal character` *`c`* `in address`

Cause: An NSAP address you entered contained a character that was not a valid hexadecimal digit.

Action: Check the NSAP address, and reenter the command line. NSAP addresses must contain only the characters 0 – 9 and a – f (or A – F).

4.10. Test Network Operation: `pingllc`

Purpose

Send LLC1 test packets to a remote node, receive the echoed response packets, compare the two packets, and report any discrepancies.

Command Line

```
pingllc MAC_address [count] [length] [subnetwork]
```

Arguments

`MAC_address`

MAC address of the remote network node.

`count`

Number of test packets to be sent. If not specified, the number defaults to 1.

`length`

Length of each test packet. If not specified, the length defaults to 1024 bytes. This utility is designed for 802.3 (Ethernet) type networks. It limits the maximum packet size to 1500 bytes.

`subnetwork`

The port number of the MAC driver attached to the subnetwork over which the test packet will be sent. The port number is specified in the STREAMS configuration file.

Notes

You can use **`pingllc`** only over a single LAN.

Command Line Example

```
pingllc 0702163200a4 20
```

Output Example

```
20 test packets sent and received
```

Error Messages

```
usage: pingllc MAC–address [count] [length] [subnetwork]
```

Cause: The command line did not contain a MAC address, or it contained too many arguments.

Action: Reenter the command line with proper arguments.

`pingllc: illegal test packet size`

Cause: The value you entered for the *`size`* argument was out of the permissible range for a MAC packet size.

Action: Reenter the command line, and specify a packet size between 1 and 1500.

`pingllc: invalid repeat count`

Cause: You entered a repeat count less than 0.

Action: Reenter the command line with a positive repeat count.

`pingllc: cannot get LLC device name`

Cause: The LLC device name does not appear in the STREAMS configuration file (`osid.cfg`).

Action: The most likely cause of this error is that the STREAMS configuration file has been corrupted or inadvertently deleted. If this is the case, you will have to restore the file from backup, or reconfigure LT–610.

`pingllc: llc1 device open failed`

Cause: The LLC1 driver is not accessible.

Action: Reinstall the LT–610 software.

`unrecognized LLC BIND ACK message`

Cause: LLC1 has sent an unexpected message type.

Action: None.

`pingllc: could not bind to a LSAP`

Cause: The LLC1 driver is too busy to handle the test request.

Action: Try again later when network traffic has decreased.

`pingllc: invalid length of node address`

Cause: The MAC address contained the wrong number of characters.

Action: Check the MAC address and reenter the corrected command line. The address must contain either 12 characters (6 bytes) or 14 characters (7 bytes).

`pingllc: illegal hexadecimal character`

Cause: a character that was not a valid hexadecimal digit was specified in the MAC address.

Action: Check the MAC address, and reenter the command line. These addresses must contain only the characters 0 – 9 and a – f (or A – F).

4.11. Configure Routing Parameters: `rtgconfig`

Purpose

Sets configuration parameters for ES–IS routing exchange protocol.

Command Line

`rtgconfig` ***`[argument value] [argument value] ...`***

Arguments

`–ct` ***`seconds`***

Sets the ES–IS configuration timer value (in seconds) for the network. This is the amount of time that can elapse before IP sends another hello packet. The value of this parameter should be set to a progressively larger value for progressively larger networks to reduce network congestion caused by ES–IS traffic. If you do not include this argument, its value defaults to 20 seconds.

`–rt` ***`seconds`***

Sets the redirect timer (in seconds). This value sets the amount of time since a remote system sent its last hello packet that its entry is maintained in the IP routing table. If you do not include this argument, its value defaults to 45 seconds. The value should always be set to at least twice the value of the configuration timer.

`–notify` ***`flag`***

Specifies whether or not the local system responds immediately to a new system coming on–line. If **`flag`** is set to a nonzero value, immediate notification is enabled; if set to zero, notification is disabled. In large networks, this notification may be disabled so that systems are not overwhelmed by notification packets as soon as they come on-line. If you do not include this argument, its value defaults to 1 (enabled).

`–esgaddr` ***`SNPA_address`***

Sets the end system multicast SNPA address recognized and used by this system. A packet sent to this address is received by all intermediate systems on the network. The SNPA address consists of 7 bytes. The first 6 bytes identify the MAC address; the seventh specifies the LSAP address (always FE). Only hexadecimal digits are accepted. Do not enter spaces or other punctuation.

NOTE: If you do not include this argument in the command line, it defaults to the correct value for a standard OSI network. This default need not normally change. If your network does not use the standard value, check the value set in the MAC driver to make sure it correlates with this one.

–isgaddr *SNPA_address*

Sets the intermediate system multicast SNPA address recognized and used by this system. A packet sent to this address is received by all end systems on the network.

NOTE: If you do not include this argument in the command line, it defaults to the correct value for a standard OSI network. This default need not normally change. If your network does not use the standard value, check the value set in the MAC driver to make sure it correlates with this one.

–help

Lists the parameters that can be set with this utility.

Notes

If no arguments are specified, this utility displays the current value of the configuration timer, redirect timer, and notify enable flag.

Command Line Example

```
rtgconfig –ct 40 –rt 60
```

Output Example

```
ES–IS Configuration timer value (seconds): 40
ES–IS Redirect timer value (seconds):      60
ES–IS Configuration Notify enable:         1
```

Error Messages

`could not open` *`device`*

Cause: The TP4/IP driver has not been installed.

Action: Reinstall LT–610 software in the kernel.

`rtgconfig: unknown parameter '`*`p`*`'`

Cause: You specified an argument on the command line that was not recognized.

Action: Check the list of arguments, and reenter the command line.

`rtgconfig: invalid SNPA–address length` *`n`*

Cause: The SNPA address contained the wrong number of characters.

Action: Check the SNPA address and reenter the corrected command line. The address must contain either 12 characters (6 bytes) or 14 characters (7 bytes).

`rtgconfig: invalid hexadecimal character` *`c`* `in address`

Cause: A character that was not a valid hexadecimal digit was specified in either the NSAP or SNPA address.

Action: Check the NSAP or SNPA address, and reenter the command line. These addresses must contain only the characters 0 – 9 and a – f (or A – F).

4.12. Display ES–IS Network Management Statistics: `rtgstat`

Purpose

Reads statistics the ES–IS routing exchange protocol maintains concerning network operation.

Command Line

`rtgstat`

Output Description

`Number of End System Hello PDU’s received`

Number of End System Hello ES–IS PDUs the system has received.

`Number of Intermediate System Hello PDU’s`

Number of Intermediate System Hello ES–IS PDUs the system has received.

`Number of Redirect PDU’s received`

Number of Redirect ES–IS PDUs the system has received.

`Number of Redirect PDU’s sent`

Number of Redirect ES–IS PDUs the system has transmitted.

`Number of Query Configuration broadcasts`

Number or Query Configuration broadcasts ES–IS PDUs the system has transmitted.

`Number of ES–IS packets discarded (any reason)`

Number of ES–IS PDUs the system has discarded for any reason.

Error Message

`could not open` *`device`*

Cause: The TP4/IP driver has not been installed.

Action: Reinstall LT–610 software in the kernel.

4.13. Load a Routing Table: `rtl`

Purpose

Load an IP routing table produced by the Network Directory Compiler.

Command Line

```
rtl file
```

Argument

`file`

The name of a file created with the Network Directory Compiler.

Notes

This utility works only on intermediate systems or on end systems that do not use ES–IS.

A Routing table is required only if ES–IS protocol is not used in the network. Refer to the *IRIS OSI Transport Service Network Directory Compiler Guide* for details.

The number of entries that the routing table can contain is limited to 64.

Error Messages

```
usage: rtl filename
```

Cause: You did not enter a filename, or you entered extraneous parameters.

Action: Reenter the command line and specify one routing table file.

```
rtl: cannot open routing table
```

Cause: The specified routing table does not exist.

Action: Check the path and filename of the routing table, then reenter the command line.

```
rtl: empty file
```

Cause: The routing table file you specified on the command line was empty.

Action: Recreate your routing table. Refer to the *IRIS OSI Transport Service Network Directory Compiler Guide*.

```
rtl: no memory
```

Cause: Memory for the routing table could not be allocated.

Action: This is generally a system problem. Refer to your system administrator’s manual.

```
rtl: read error
```

Cause: The routing table file is corrupted.

Action: Recreate your routing table. Refer to the *IRIS OSI Transport Service Network Directory Compiler Guide*.

4.14. Download a DTE/Port Mapping Table: `sndcmap`

Purpose

This utility downloads a mapping table to the SNDCF driver which allows the called DTE address in an outgoing call request to be associated with a local port number.

Command Line

`sndcmap -V` ***`file`***

Argument

`–V`

Prints version number of **`sndcmap`**. Selecting this option will cause **`sndcmap`** to exit without downloading the mapping table.

`–s`

Specifies, in bytes, the size of the output table that will hold the mapping information. If this argument is not specified, the default value is 4K.

`file`

Specifies the input file which contains the mapping information. Figure 4-1 shows an example input file. The # sign indicates a comment.

```
#  called dte address   local port
   3259002918           0
   12 *                 0
   7298765432222        1
```

Figure 4-1. Sample `sndcmap` Input File

Ports are specified using the convention described in Table 3-1, on page 3-4. If you are only using one card, then Ports 2 and 3 are not relevant. The default is to choose any available port.

The asterisk wildcard character, **`*`**, (separated by a space from the remainder of the address) may be used to specify all DTE's beginning with certain values: for example, **`333 *`** refers to all DTEs beginning with **`333`**. All uneven DTE addresses must be padded with **`f`**.

Use of the **`sndcmap`** utility is only required if it is necessary to route requests to a particular port. If, for example, all ports on the card(s) being used are associated with the same network, use of **`sndcmap`** is not necessary, though it may still be used if wished. If no **`sndcmap`** input file is downloaded, SNDCF simply uses any available port on which to send the call requested.

4.15. Configure TP4 Parameters: `tp4config`

Purpose

Sets the value for the specified TP4 configuration parameters, then displays all parameter values. Note that all times are specified in milliseconds.

Command Line

`tp4config` ***`[–arg value] [–arg value] ...`***

Notes

With all arguments that specify time, the value supplied is rounded up to the next multiple of 10 to ensure granularity with the system clock. The arguments in question are:**`–inactivity`**, **`–retransmit`**, **`–upper`**, **`–lower`**, **`–window`**, **`–ack`**, and **`–flow`**.

Arguments

`–inactivity` ***`time`***

The amount of time that TP4 maintains a transport connection without any data transmission before it terminates the connection. If this argument is not specified, it defaults to the value derived from the product of the current retransmit count (**`–n`** argument) and the window size (**`–window`** argument).

`–retransmit` ***`time`***

The amount of time that TP4 initially waits without receiving an acknowledgment of a TPDU it sent before it retransmits the TPDU. (The maximum number of times the same TPDU can be transmitted can be specified with the **`–n`** option.) TP4 automatically adjusts the value of this parameter individually for each connection based on the performance of the network. If this argument is not specified, it defaults to 2000 (2 seconds).

`–upper` ***`time`***

Upper bound of the retransmit timer; the largest value to which TP4 can set the retransmit time. If this argument is not specified, it defaults to 10,000 (10 seconds).

`–lower` ***`time`***

Lower bound of the retransmit timer; the smallest value to which TP4 can set the retransmit time. If this argument is not specified, it defaults to 10 (10 milliseconds).

–window *time*

The period of inactivity that can occur before TP4 sends a hello packet to the peer entity with which a connection is established. If you do not include this argument, its value defaults to 20,000 (20 seconds).

–ack *time*

Local acknowledgment delay; the maximum amount of time TP4 can delay acknowledgment of a TPDU it receives. If you do not include this argument, its value defaults to 100 (0.1 seconds).

–flow *time*

If TP4 has no receive credit available, it checks the available resources every **time** milliseconds until credit becomes available. If you do not include this argument, its value defaults to 100 (0.1 seconds).

–n *iterations*

Maximum number of times a TPDU can be sent without being acknowledged. If you do not include this argument, its value defaults to 9.

–credit *value*

Maximum value to which TP4 can adjust the window size; that is, the number of unacknowledged TPDUs that can be outstanding on a connection. If you do not include this argument, its value defaults to 4.

–initial *value*

The initial window size when the connection is first established. TP4 will adjust this value up to the maximum automatically. If you do not include this argument, its value defaults to 1.

–maxtpdu *value*

Maximum size of a TPDU. It is recommended that you select one of the following values: 128, 256, 512, 1024, or 2048 bytes. If you do not include this argument, its value defaults to 1024. If you change the default, it is important that you ensure there are sufficient STREAMS buffers of the appropriate size. Use the `strstat` utility to determine the number of buffers available in the size you specify, and increase this number if necessary using the procedure given in the LT–610 installation appendix.

–checksum *flag*

If **flag** is a nonzero value, the TPDU checksum feature is enabled; if zero, it is disabled. Enabling this feature slows down TP4 operation significantly. If you do not include this argument, its value defaults to 0 (disabled).

–propose *flag*

If **`flag`** is a nonzero value, Transport proposes to use extended TPDU sequencing when it initiates a connect request (CR) TPDU; if zero, Transport proposes normal sequencing. If you do not include this argument, its value defaults to 1 (enabled).

–decca *flag*

If **`flag`** is a nonzero value, the DEC congestion avoidance flag is enabled; if zero, the flag is disabled. If you do not include this argument, its value defaults to 0 (disabled).

Keep in mind that the DEC congestion avoidance algorithm is effective only if all end and intermediate systems in a network use it. Refer to the National Institute of Standards and Technology (NIST) Special Publication 500–162 for details on this algorithm.

Command Line Example

```
tp4config –n 15 –upper 25000 –lower 50
```

Output Example

```
Inactivity time:              400000
Initial retransmit timer:2000
Upper bound on retransmit timer: 25000
Lower bound on retransmit timer:50
Window timer value:20000
Local acknowledgment delay:100
Local flow control interval:100
Max number of transmissions:10
Target window size:4
Initial window size:1
Maximum TPDU size:1024
Maximum TIDU size: 1016
Checksum enable:0
Prefer use of extended sequence numbers:
Propose use of extended sequence numbers:11
DEC Congestion avoidance enable:0
Max number of transport connections (read/only):32
```

Error Messages

`could not open` *`device`*

Cause: The TP4/IP driver has not been installed.

Action: Reinstall LT–610 software in the kernel.

`tp4config: unknown parameter` *`p`*

Cause: You specified an argument on the command line that was not recognized.

Action: Check the list of arguments and reenter the command line.

4.16. Read TP4 Network Management Statistics: `tp4stat`

Purpose

Read the various statistics TP4 keeps concerning network operation.

Command Line

`tp4stat`

Output Description

`Number of open and established connections`

Number of connections currently open and in the data transfer phase at this system.

`Total number of TPDU’s transmitted`

Number of TPDUs sent from this system. This number includes retransmitted TPDUs.

`Total number of TPDU’s received`

Number of correctly formatted TPDUs this system has received.

`Number of DT TPDU’s transmitted`

Number of complete Data Transfer (DT) TPDUs this system has transmitted. This number does not include retransmitted TPDUs.

`Number of DT TPDU’s received`

Number of complete Data Transfer (DT) TPDUs this system has received.

`Number of AK TPDU’s transmitted`

Number of Acknowledge (AK) TPDUs this system has transmitted, including duplicated AK TPDUs.

`Number of AK TPDU’s received`

Number of Acknowledge (AK) TPDUs this system has received, including duplicates.

`Number of ED TPDU’s transmitted`

Number of Expedited Data (ED) TPDUs this system has transmitted. This count does not include retransmitted ED TPDUs.

`Number of ED TPDU’s received`

Number of Expedited Data (ED) TPDUs this system has received.

`Number of UD TPDU’s transmitted`

Number of Unit Data (UD) TPDUs this system has transmitted.

`Number of UD TPDU’s received`

Number of Unit Data (UD) TPDUs this system has received.

`Total number of TPDU’s retransmitted`

Number of retransmitted Connect Request (CR), Connect Confirm (CC), Data Transfer (DT), Expedited Data (ED), and Disconnect Request (DR) TPDUs this system has retransmitted.

`Number of DT TPDU’s retransmitted`

Number of Data Transfer (DT) TPDUs this system has retransmitted.

`Number of ED TPDU’s retransmitted`

Number of Expedited Data (ED) TPDUs this system has retransmitted.

`Number of receive window closures`

Number of credit reductions in which the window is closed after the reduction.

`Number of credit reductions`

Number of credit reductions in which the window remains open after the reduction.

`Number of duplicate DT TPDU’s received`

Number of duplicate Data Transfer (DT) TPDUs this system has received and discarded.

`Number of duplicate ED TPDU’s received`

Number of duplicate Expedited Data (ED) TPDUs this system has received and discarded.

`Number of out of window DT TPDU’s rcvd`

Number times this system has received a Data Transfer (DT) TPDU with a sequence number higher than the upper window edge.

`Number of out of sequence AK TPDU’s rcvd`

Number of times this system has received an out–of–sequence Acknowledge (AK) TPDU. This counter does not include duplicate AK TPDUs.

`Successfully established incoming connections`

Number of times this system has received a Connect Request (CR) TPDU that resulted in an established connection.

`Successfully established outgoing connections`

Number of times this system has transmitted a Connect Request (CR) TPDU that resulted in an established connection.

`Number of rejected incoming connections`

Number of times this system has received a Connect Request (CR) TPDU that did not result in an established connection.

`Number of rejected outgoing connections`

Number of times this system has transmitted a Connect Request (CR) TPDU that did not result in an established connection.

`Number of user rejected incoming connections`

Number of times a TSU of this system has rejected an incoming connection.

`Incoming CR TPDU’s rejected due to configuration`

Number of times this system has received a Connect Request (CR) TPDU, and responded to it with a Disconnect Request (DR) TPDU. This counter is incremented when the Transport reason code returned is 2, 3, 130, 131, 132, 135, or 136.

`Outgoing CR TPDU’s rejected due to configuration`

Number of times the system has received a Disconnect Request (DR) TPDU in response to a Connection Request (CR) TPDU it transmitted. This counter is incremented when the Transport reason code returned is 2, 3, 130, 131, 132, 135, or 136.

`Incoming CR TPDU’s rejected due to protocol error`

Number of times the system has received a TPDU and rejected it because it contained a protocol error. This counter is incremented when the Transport reason code returned is 0, 133, or 138.

`Outgoing CR TPDU’s rejected due to protocol error`

Number of times the system received a Disconnect Request (DR) or Error Report (ER) TPDU in response to a Connect Request (CR) TPDU. This counter is incremented when the Transport reason code returned is 0, 133, or 138.

`Number of TPDU transmission timeouts`

Number of instances in which the system has retransmitted a TPDU the maximum number of times without receiving any response.

`Number of inactivity timeouts`

Number of times a Transport state machine in the system has timed out because no TPDUs were received on the connection.

`Number of received TPDU’s causing a protocol error`

Number of TPDUs the system receives which contain a protocol error and cause the Transport protocol to transmit a Disconnect Request (DR) TPDU. This TPDU will contain the Transport reason code 133.

`Number of DR or ER TPDU’s received indicating a protocol error`

Number of times the system has received a Disconnect Request (DR) or Error Report (ER) TPDU indicating that a TPDU that the system transmitted previously (other than a CR TPDU) contained a protocol error.

`Number of TPDU’s with invalid checksum`

Number of times the system discarded a TPDU because the checksum indicated that the TPDU was bad.

`Number of received TPDU’s ignored (bad header)`

Number of times the system received a TPDU having an incorrect header.

`Number of received and ignored UD TPDU’s`

Number of times the system received a Unit Data (UD) TPDU and discarded it for any reason.

Error Messages

`could not open` *`device`*

Cause: The TP4/IP driver has not been installed.

Action: Reinstall LT–610 software in the kernel.

5. WT-325 UTILITIES

This chapter discusses the network management utilities provided with the WT–325 product. These utilities set and display the values of configuration parameters and manage routing tables.

NOTE: All WT–325 Network Management utilities require superuser privileges.

Contents

5.1. Introduction

This chapter describes the WT–325 utility programs you can use to adjust various aspects of the protocol stack after you use the **`osid`** program to configure it.

5.2. Download DTE Mapping: `dtemap`

Purpose

This utility downloads a mapping table to the TP0/2/4 driver which allows a called NSAP to be mapped to a DTE address (required by the X.25 protocol software on the WAN card) and an associated port on the card.

Command Line

```
dtemap [–V] [–s] file
```

Arguments

`–V`

Prints version number of **`dtemap`**. Selecting this option will cause **`dtemap`** to exit without downloading the DTE mapping table.

`–s`

Specifies, in bytes, the size of the output table that will hold the mapping information. If this argument is not specified, the default value is 4K.

`file`

Specifies the input file which contains the mapping information. Figure 5-1 shows an example input file.

```
#  remote         remote              local
#  nsap           dte address         port
      -           3259 *              0
      -           1289 *              3
      -           12345678902222      1
   3259 *         3259002918          0
   #471322323232  2132222f            2
   #372147328722  244f                3
   #12 *          12345678902f        2
```

Figure 5-1. Sample `dtemap` Input File

The **`dtemap`** input file can contain entries with 2 different formats.

1. If the application running over WT–325 uses DTE addresses, then the **`nsap`** entry is left blank, and only the associated port is specified. For example, if you have one WAN card installed, and therefore 2 ports, then for each possible called DTE address that may be used you specify either Port 0 or 1.
2. If the application running over WT–325 uses NSAP addresses, an NSAP / DTE address / port combination is used. For example, if a remote NSAP is to be included in the **`dtemap`** input file, you must also specify the corresponding called DTE address and the local port on which the call will go out.

Ports are specified using the convention described in Table 3-1, on page 3-4. The default is to choose any available port. If no **`dtemap`** input file is downloaded, TP0/2/4 simply uses any port on which to send the call requested.

The asterisk wildcard character, **`*`**, may be used to specify all DTEs beginning with certain values: for example, **`333 *`** refers to all DTEs beginning with **`333`**. Note that a space separates the **`333`** and the **`*`**.

NOTE: All uneven DTE addresses must be padded with **`f`**.

5.3. Configure TP0/2/4 Parameters: `tp24config`

Purpose

Sets the value for the specified TP0/2/4 configuration parameters. If no argument is supplied, **`tp24config`** displays all parameter values. Note that all times are specified in milliseconds.

Command Line

```
tp24config [–arg value] [–arg value] ...
```

Arguments

`–inactivity` ***`time`***

(Class 4 only) Inactivity time; the amount of time that TP0/2/4 maintains a transport connection without any data transmission before it terminates the connection. If this argument is not specified, it defaults to 960,000 (960 seconds).

`–retransmit` ***`time`***

(Class 4 only) The amount of time that TP0/2/4 initially waits without receiving an acknowledgment of a TPDU it sent before it retransmits the TPDU. (The maximum number of times the same TPDU can be transmitted can be specified with the **`–n`** option.) TP0/2/4 automatically adjusts the value of this parameter individually for each connection based on the performance of the network. If this argument is not specified, it defaults to 30,000 (30 seconds).

`–window` ***`time`***

(Class 4 only) Window timer value; the period of inactivity that can occur before TP0/2/4 sends an “Ack” packet to the peer entity with which a connection is established. If this argument is not specified, its value defaults to 120,000 (120 seconds).

`–ack` ***`time`***

(Class 4 only) Local acknowledgment delay; the maximum amount of time TP0/2/4 can delay acknowledgment of a TPDU it receives. If this argument is not specified, its value defaults to 512 (0.512 seconds).

`–flow` ***`time`***

If TP0/2/4 has no receive credit available, it checks the available resources every **`time`** milliseconds until credit becomes available. If this argument is not specified, its value defaults to 208 (0.208 seconds).

`–n` ***`iterations`***

(Class 4 only) Maximum number of times a TPDU can be sent without being acknowledged. If this argument is not specified, its value defaults to 4.

–credit *value*

(Classes 2 and 4 only) Maximum value to which TP0/2/4 can adjust the window size; that is, the number of unacknowledged TPDUs that can be outstanding on a connection. If this argument is not specified, its value defaults to 4.

–initial *value*

(Class 4 only) The initial window size when the connection is first established. TP0/2/4 will adjust this value up to the maximum automatically. If this argument is not specified, its value defaults to 1.

–maxtpdu *value*

Maximum size of a TPDU. It is recommended that you select one of the following values: 128, 256, 512 or 1,024 bytes. If this argument is not specified, its value defaults to 1,024. If you change the default, it is important that you ensure there are sufficient STREAMS buffers of the appropriate size. Use the `strstat` utility to determine the number of buffers available in the size you specify, and increase this number if necessary using the procedure given in the WT–325 installation appendix. If this argument is not specified, its value defaults to 512.

–maxtidu *value*

Maximum number of bytes of the data units passed between the TLI library and the TP0/2/4 STREAMS driver. If this is larger than the TPDU size, the STREAMS driver will fragment the message into TPDU–size packets before sending it to Transport. If this argument is not specified, its value defaults to 504.

–checksum *flag*

(Class 4 only) If **`flag`** is a nonzero value, the TPDU checksum feature is enabled; if zero, it is disabled. Enabling this feature slows down TP0/2/4 operation significantly. If this argument is not specified, its value defaults to 0 (disabled).

–extended *flag*

(Class 4 only) If **`flag`** is a nonzero value, Transport proposes to use extended TPDU sequencing when it initiates a connect request (CR) TPDU; if zero, Transport proposes normal sequencing. If this argument is not specified, its value defaults to 1 (enabled).

–propose *flag*

(Class 4 only) If **`flag`** is a nonzero value, Transport proposes to use extended TPDU sequencing when it initiates a connect request (CR) TPDU; if zero, Transport proposes normal sequencing. If this argument is not specified, its value defaults to 1 (enabled).

–transaction ***time***

Number of milliseconds that a network connection is held open waiting for an ISO 8602 UD TPDU. (ISO 8602 connectionless transport protocol only.) If this argument is not specified, its value defaults to 60,000 (60 seconds).

–ts1 ***time***

(Classes 0 and 2 only) The maximum amount of time that Transport will wait for a response after a CR TPDU has been transmitted. If this argument is not specified, it value defaults to 60,000 (60 seconds).

–ts2 ***time***

(Classes 0 and 2 only) The maximum amount of time that Transport will wait for a response after a DR TPDU has been transmitted. If this argument is not specified, its value defaults to 30,000 (30 seconds).

–delay ***time***

The time to delay before disconnecting the network connection. The purpose of this small delay is to avoid loss of data during network layer disconnection. If this argument is not specified, its value defaults to 1000 (1 second).

–idle ***time***

The maximum time which an idle network connection will be left open. If this argument is not specified, its value defaults to 300,000 (300 seconds).

–useflow ***flag***

If **flag** is a zero value, Transport attempts to negotiate non–use of explicit flow control. If you do not include this argument, its value defaults to 1 (forces Class 2 to use explicit flow control).

–ccitt ***flag***

If **flag** is a nonzero value, Transport follows CCITT conformance rules in regard to the classes proposed in CR TPDU's. If either Class 2 or 4 is requested by the user, Class 0 is proposed as an alternate class. If **flag** is zero, no alternate Class parameter is transmitted in a CR TPDU.

Command Line Example

```
tp24config
```

Output Example

```
 NOTE: All timer values specified in number of milliseconds
Reference timer value (class 4 only):              400000
Inactivity timer value (class 4 only):             960000
Retransmit timer value (class 4 only):              30000
```

```
Window timer value (class 4 only):                    120000
Local acknowledgement delay (class 4 only):              512
Local flow control interval (all classes):               208
Max number of transmissions (class 4 only):                4
Target window size (class 2 and 4):                        4
Initial window size (class 4 only):                        1
Maximum TPDU size (all classes):                         512
Maximum TIDU size (all classes):                         504
Checksum enable (class 4 only):                            0
Prefer use of extended sequence numbers
(class 2 and 4):                                           1
Propose use of extended sequence numbers
(class 2 and 4)                                            1
Disconnection timer value (CLTS only):                120000
Optional TS1 timer (class 0/2):                        60000
Optional TS2 timer (class 2):                          60000
Disconnection timer value (class 2 and 4):              2000
Incoming network connection idle timer:               300000
Use of explicit flow control (class 2 only):               1
CCITT 1988 X.224 alternate class 0 enable
(class 2 and 4):                                           1
Max number of transport connections (read/only):          48
```

Error Messages

`could not open` *`device`*

Cause: The TP0/2/4 driver has not been installed or the **`osid`** utility is not running.

Action: Make sure that the protocol stack has been initialized. Stop and start the OSI Transport Service, using the instructions in section1.7, page 1-14. If the stack cannot be started because of missing drivers, then reinstall WT–325 software in the kernel.

`tp24config: unknown parameter` *`p`*

Cause: You specified an argument on the command line that was not recognized.

Action: Check the list of arguments, and reenter the command line.

5.4. Read TP0/2/4 Network Management Statistics: `tp24stat`

Purpose

Reads the various statistics TP0/2/4 keeps concerning network operation.

Command Line

```
tp24stat
```

Output Description

`Number of open and established connections`

Number of connections currently open and in the data transfer phase at this system.

`Total number of TPDU’s transmitted`

Number of TPDUs sent from this system. This number includes retransmitted TPDUs.

`Total number of TPDU’s received`

Number of correctly formatted TPDUs this system has received.

`Number of DT TPDU’s transmitted`

Number of complete Data Transfer (DT) TPDUs this system has transmitted. This number does not include retransmitted TPDUs.

`Number of DT TPDU’s received`

Number of complete Data Transfer (DT) TPDUs this system has received.

`Number of AK TPDU’s transmitted`

Number of Acknowledge (AK) TPDUs this system has transmitted, including duplicate AK TPDUs.

`Number of AK TPDU’s received`

Number of Acknowledge (AK) TPDUs this system has received, including duplicates.

`Number of ED TPDU’s transmitted`

Number of Expedited Data (ED) TPDUs this system has transmitted. This count does not include retransmitted ED TPDUs.

`Number of ED TPDU’s received`

Number of Expedited Data (ED) TPDUs this system has received.

`Number of UD TPDU’s transmitted`

Number of Unit Data (UD) TPDUs this system has transmitted.

`Number of UD TPDU’s received`

Number of Unit Data (UD) TPDUs this system has received.

`Total number of TPDU’s retransmitted`

Number of retransmitted Connect Request (CR), Connect Confirm (CC), Data Transfer (DT), Expedited Data (ED) , and Disconnect Request (DR) TPDUs this system has retransmitted.

`Number of DT TPDU’s retransmitted`

Number of Data Transfer (DT) TPDUs this system has retransmitted.

`Number of ED TPDU’s retransmitted`

Number of Expedited Data (ED) TPDUs this system has retransmitted.

`Number of receive window closures`

Number of credit reductions in which the window is closed after the reduction.

`Number of credit reductions`

Number of credit reductions in which the window remains open after the reduction.

`Number of duplicate DT TPDU’s received`

Number of duplicate Data Transfer (DT) TPDUs this system has received and discarded.

`Number of duplicate ED TPDU’s received`

Number of duplicate Expedited Data (ED) TPDUs this system has received and discarded.

`Number of out of window DT TPDU’s rcvd`

Number of times this system has received a Data Transfer (DT) TPDU with a sequence number higher than the upper window edge.

`Number of out of sequence AK TPDU’s rcvd`

Number of times this system has received an out–of–sequence Acknowledge (AK) TPDU. This counter does not include duplicate AK TPDUs.

`Successfully established incoming connections`

Number of times this system has received a Connect Request (CR) TPDU that resulted in an established connection.

`Successfully established outgoing connections`

Number of times this system has transmitted a Connect Request (CR) TPDU that resulted in an established connection.

`Number of rejected incoming connections`

Number of times this system has received a Connect Request (CR) TPDU that did not result in an established connection.

`Number of rejected outgoing connections`

Number of times this system has transmitted a Connect Request (CR) TPDU that did not result in an established connection.

`Number of user rejected incoming connections`

Number of times a TSU of this system has rejected an incoming connection.

`Incoming CR TPDU’s rejected due to configuration`

Number of times this system has received a Connect Request (CR) TPDU, and responded to it with a Disconnect Request (DR) TPDU. This counter is incremented when the transport reason code returned is 2, 3, 130, 131, 132, 135 or 136.

`Outgoing CR TPDU’s rejected due to configuration`

Number of times the system has received a Disconnect Request (DR) TPDU in response to a Connection Request (CR) TPDU it transmitted. This counter is incremented when the transport reason code returned is 2, 3, 130, 131, 132, 135 or 136. The most likely cause of this error is that no TSU is bound to the address received in the connect request PDU.

`Incoming CR TPDU’s rejected due to protocol error`

Number of times the system has received a TPDU and rejected it because it contained a protocol error. This counter is incremented when the transport reason code returned is 0, 133, or 138.

`Outgoing CR TPDU’s rejected due to protocol error`

Number of times the system received a Disconnect Request (DR) or Error Report (ER) TPDU in response to a Connect Request (CR) TPDU. This counter is incremented when the transport reason code returned is 0, 133, or 138.

`Number of TPDU transmission timeouts`

Number of instances in which the system has retransmitted a TPDU the maximum number of times without receiving any response.

`Number of inactivity timeouts`

Number of times a transport state machine in the system has timed out because no TPDUs were received on the connection.

`Number of received TPDUs causing a protocol error`

Number of TPDUs the system receives which contain a protocol error and cause the transport protocol to transmit a Disconnect Request (DR) TPDU. This TPDU will contain the transport reason code 133.

`Number of DR or ER TPDU received indicating a protocol error`

Number of times the system has received a Disconnect Request (DR) or Error Report (ER) TPDU indicating that a TPDU that the system transmitted previously (other than a CR TPDU) contained a protocol error.

`Number of TPDU’s with invalid checksum`

Number of times the system discarded a TPDU because the checksum indicated that the TPDU was bad.

`Number of received TPDU’s ignored (bad header)`

Number of times the system received a TPDU with an incorrect header.

`Number of received and ignored UD TPDUs`

Number of times the system received a Unit Data (UD) TPDU and discarded it for any reason.

`Number of incoming network connects with invalid protocol ID`

The number of times that a packet is received by Transport and the CALL user data field (protocol identification) does not indicate either ISO 8073 or ISO 8602.

`Number of network layer reset indications`

Number of times the network layer was reset.

`Number of transport TS1/TS2 timeouts`

Number of timeouts following transmission of a CR TPDU or a DR TPDU.

`Number of network layer abnormal disconnects`

Number of network layer disconnects (received by Transport) which cause an abnormal transport layer disconnect.

Error Messages

`could not open` *`device`*

Cause: the TP0/2/4 driver has not been installed or the **`osid`** utility is not running.

Action: make sure that the protocol stack has been initialized Stop and start the OSI Transport Service, using the instructions in section1.7, page 1-14. If the stack cannot be started because of missing drivers, then reinstall WT–325 software in the kernel.

A. HOW TO TEST THE SOFTWARE INSTALLATION

Contents

A.1. Introduction

After the OSI Transport Service software is installed, it should be tested with a minimally functional application to ensure that both software and supporting hardware are fully operational. The OSI Transport Service product contains two sample test programs to accomplish this testing: **`sttxt`** and **`strxt`**.

This chapter summarizes how to test minimal communications between two systems using **`sttxt`** and **`strxt`** over a simulated or a real X.25 network (using the Transport Class 0/2/4/X.25 PLP/LAPB protocol stack). Once you are confident that the OSI Transport Service software is installed correctly, you can try communicating over a real network.

A.2. Materials Required for Achieving Communications Between Systems

The minimal configuration required to demonstrate LAN or WAN card operation is shown in Figure A-1. The required items are:

- Two host computers each having an installed LAN or WAN card.
- Appropriate interconnection cables.

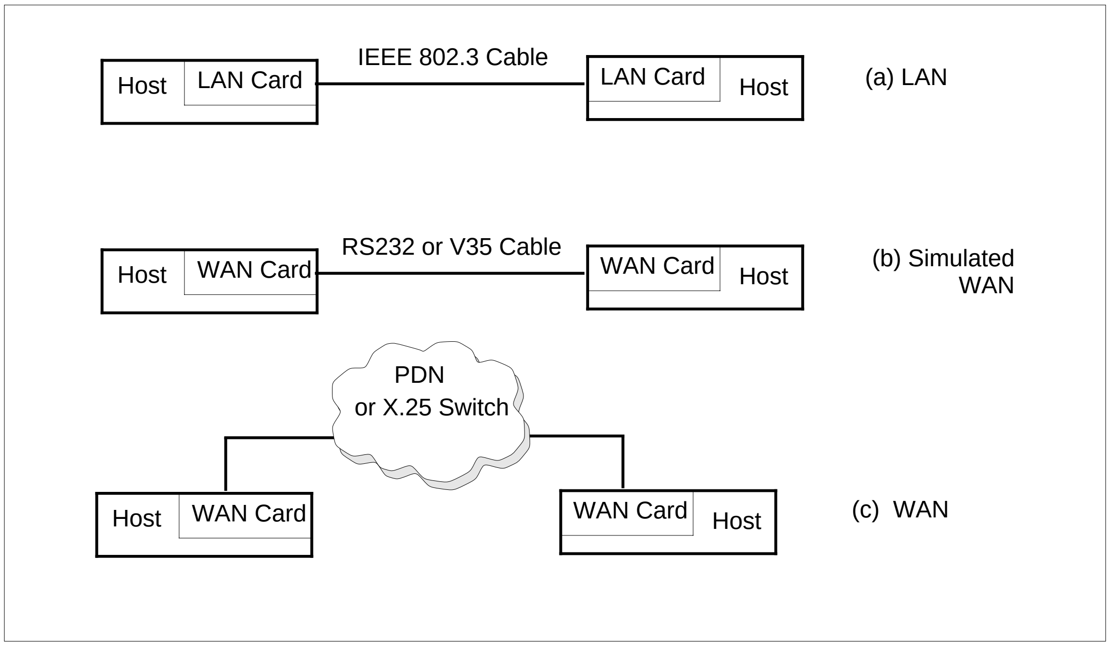


Figure A-1. Configurations for Testing LAN and WAN Installation

A.3. Testing OSI Transport Service Software Installation

The **`sttxt`** and **`strxt`** utilities are used as the receiving and transmitting ends of a dual system test. They work together to ensure that OSI lower layers on communicating systems have been set up and configured correctly.

A.3.1. Transmission Testing Utility: `sttxt`

Purpose

Activates the transmitting end of a dual–system test. The basic function of the utility is to set up a transport endpoint, bind to that endpoint, and then issue a connect request. Once a connection is set up, **`sttxt`** will send data TPDUs to the receiving end of the connection (typically **`strxt`**).

NOTE: If **`sttxt`** is used without arguments, the default is Class 4, CLNS: that is, LAN.

Command Line

```
sttxt [–arg [value]] [–arg [value]] ...
```

Arguments

`0`

Specifies TP Class 0 and CONS.

`2`

Specifies TP Class 2 and CONS.

`4`

Specifies TP Class 4 and CONS.

`x`

Enables debugging for printing verbose messages during the execution of **`sttxt`**.

`b value`

Specifies the number of bytes of data in each transmitted TPDU. The **`strxt`** and **`sttxt`** utilities should agree on the number of bytes in exchanged TPDUs. The default value is 16,256 bytes.

`c value`

Specifies the maximum number of connections.

`d`

Enables data integrity checking. For example, specifying **`-d123456`** indicates that the sequence "123456" be used as the data content of transmitted TPDUs. If the same value is specified on the **`strxt`** side of the connection, you can verify that data is transmitted uncorrupted. The default is to transmit zeroes.

`n value`

Specifies the remote NSAP or DTE address.

`q`

Enables quiet mode. Only error messages are displayed.

`r value`

Specifies Responder TSAP selector.

`s value`

Specifies the number of seconds between statistics reports.

`v`

Enables varying of TIDU size.

`V`

Prints the version number of **`sttxt`**. Choosing this switch will disable **`sttxt`** from executing.

A.3.2. Reception Testing Utility: `strxt`

Purpose

Activates the receiving end of a dual–system test. The basic function of the utility is to set up a transport endpoint, bind to that endpoint, and then listen for incoming connect requests. Once a connection is set up, **`strxt`** will receive incoming data from the transmitting end of the connection.

NOTE: If **`strxt`** is used without arguments, the default is Class 4, CLNS: that is, LAN.

Command Line

```
strxt [–arg [value]] [–arg [value]] ...
```

Arguments

`0`

Specifies TP Class 0 and CONS.

`2`

Specifies TP Class 2 and CONS.

`4`

Specifies TP Class 4 and CONS.

`b value`

Specifies expected number of bytes of data in each TPDU. The **`strxt`** and **`sttxt`** utilities should agree on the number of bytes in exchanged TPDUs. The default value is 16,256 bytes.

`d`

Enables data integrity checking. For example, specifying **`-d123456`** indicates that the sequence "123456" be used to check the data content of incoming TPDUs. If the same value is specified on the **`sttxt`** side of the connection, you can verify that data is transmitted uncorrupted. The default is to expect zeroes.

`q`

Enables quiet mode. Only error messages are displayed.

`r value`

Specifies Responder TSAP selector.

`s value`

Specifies the number of seconds between statistics reports.

`v`

Enables varying of packet size.

`V`

Prints the version number of **`strxt`**. Choosing this switch will disable **`strxt`** from executing.

A.4. Setting Up and Running Tests

When installation is complete, you can verify OSI Transport Service operation. To do this, run the connectivity verification utilities **`strxt`** and **`sttxt`**. The test can be run using two systems over a simulated or real LAN or WAN.

A.4.1. Testing LAN Connectivity

Perform the following tests to establish LAN connectivity:

1. Log in to the receiving system.
2. Start the OSI protocol stack. Enter:

   ```
   osid -v
   ```

3. Use the **`net`** and **`nsap`** commands to add a network title and specify an NSAP for the protocol stack. Consult section 4.8, page 4-15, and section 4.9, page 4-17, for information on how to use these utilities.
4. Start the receiving end of the test. Enter:

   ```
   strxt –b100
   ```

5. Log in to the transmitting system.
6. Start the OSI protocol stack. Enter:

   ```
   osid -v
   ```

7. Enter:

   ```
   sttxt –b100 –n NSAP
   ```

   Replace *`NSAP`* with the NSAP for the remote system.
8. Messages indicating connection establishment should appear on the screens of both systems immediately. Thereafter, a throughput measurement message should appear every 30 seconds on both screens.

A.4.2. Testing WAN Connectivity

Perform the following tests to establish WAN connectivity:

1. Ensure that the configuration file, `x25cfg333.2` has been edited correctly for each end system (consult the IRIS X.25 protocol software documentation). This involves setting the port(s) to be used to type DTE, and filling in the DTE address(es).
2. If systems are not joined by a Public Data Network or other switch environment that supplies DCE operation, then change the port type on one of the end systems to DCE. If the WAN software has been configured to use both ports on the card (i.e., there are two **`MX25_n`** entries in `/etc/osid.cfg`), then set both ports to DCE on the end system, and check the DTE addresses. See Table 3-1 on page 3-4 for a description of the correct use of ports.
3. Log in to the receiving system.
4. Start the OSI protocol stack. Enter:

   ```
   osid -v
   ```

5. Start the receiving end of the test. Enter:

   ```
   strxt –0 –b100
   ```

6. Log in to the transmitting system.
7. Start the OSI protocol stack. Enter:

   ```
   osid -v
   ```

8. Enter:

   ```
   sttxt –0 –b100 –n DTE_address
   ```

   Replace ***`DTE_address`*** with the DTE address for the remote system.

9. Messages indicating connection establishment should appear on the screens of both systems immediately. Thereafter, a throughput measurement message should appear every 30 seconds on both screens.
10. When you are convinced that the test is working, halt the test by typing `<Ctrl>+<C> in` both systems. You should then run the responder/initiator on the opposite systems to test the connectivity in the opposite direction.

B. SAMPLE NDC SOURCE FILE

Contents

B.1. Introduction

The IRIS Network Directory Compiler (NDC) can generate routing table object files for use by the CLNP network layer protocol. The NDC takes as its input files that define the network topology, called network description source files. This appendix presents a sample network description source file.

The network topology is shown in Figure B-1. There are three intermediate systems (**`achil`**, **`aran`** and **`bofin`**) linking four subnetworks (**`ether1`**, **`ether2`**, and **`ether3`** are 802.3 LANs, while **`pdn`** is an X.25 WAN). **`achil`** and **`aran`** are both connected to the **`pdn`** subnetwork through Port 0 on their WAN cards.

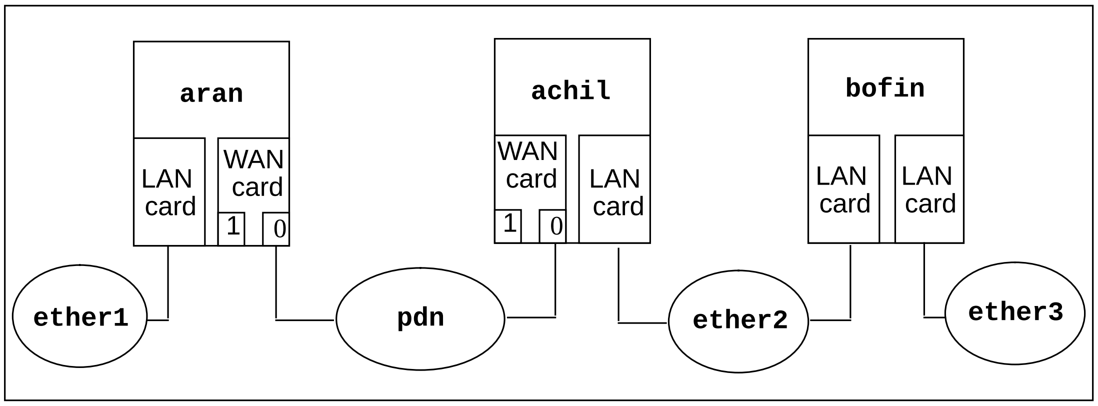


Figure B-1. Sample Network Topology

The network description source file describing this network is shown in section B.2, page B-3. For a complete description of the network description file syntax, consult the *IRIS OSI Transport Service Network Directory Compiler Guide*.

B.2. Sample NDC Source File

```
BEGIN

SUBNETS

ether1: /49.0004.0006.0020 - /49.0004.0006.0021    [14 - 25,254]

ether2: /49.0004.0006.0021 - /49.0004.0006.0022    [14 - 25,254]

ether3: /49.0004.0006.0022 - /49.0004.0006.0023    [14 - 25,254]

pdn:    /49.0004 - /49.0005                        [14 - 25,254]

SYSTEMS

aran:   {

                pdn = 1,/3259002917

                ether1 = 0,/0800690617DAFE,ESIS

        }

achil:  {

                pdn = 1,/3259002919

                ether2 = 0,/080069021B9FFE,ESIS

        }

bofin:  {

                ether2 = 0,/080069061667FE,ESIS

                ether3 = 1,/02CF1F114961FE,ESIS

        }

DIRECTORIES

SYSTEMS

bofin = bofin

aran = aran

achil = achil

END
```

B.3. Elements of the Sample NDC Source File

The major elements in the NDC source file are:

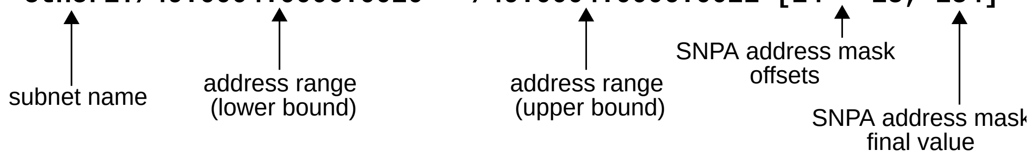
- Subnets
- Systems
- Directories

SUBNETS

The address ranges for each known subnetwork are described including the authority and format identifier (AFI), initial domain identifier (IDI), organization ID, subnet ID. Following each address range in this case is an SNPA address mask.

For example:

```
ether1:/49.0004.0006.0020 - /49.0004.0006.0021 [14 - 25, 254]
```


subnet name
address range (lower bound)
address range (upper bound)
SNPA address mask offsets
SNPA address mask final value

The address range is inclusive at the lower bound, but exclusive at the upper bound. The period (`.`) character in all address ranges is ignored.

The SNPA (SubNetwork Point of Attachment) address mask specifies how to extract the SNPA address of the remote system from the complete NSAP of the remote system. It consists of offsets to the start and finish of the SNPA address, and optionally a decimal fixed value to append (in hexadecimal form) to the end of the extracted digits.

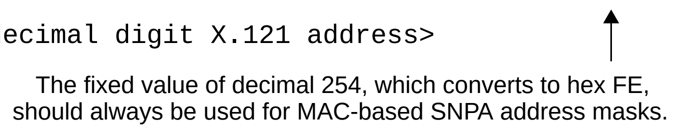
For example, **`[14 - 25,254]`** will convert **`49000400000080800890617DA01`** into **`0800690617FE`**, as follows:

"Digits from the 14th position (inclusive) to the 25th position (exclusive) are used, and the hexadecimal equivalent of 254, that is, FE, is appended."

An SNPA address can be MAC-based or X.121-based:

Mac based: `<12 Hex digit MAC address of LAN Card><LSAP(hex FE)>`

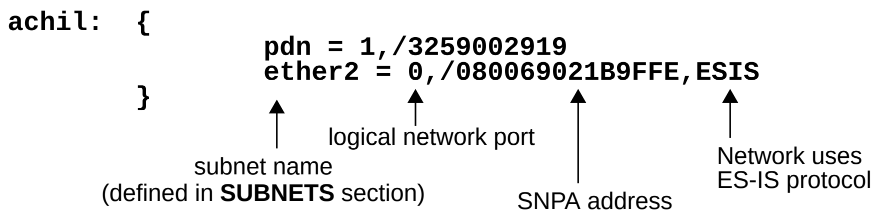
X.121 based: `<Up to 15 decimal digit X.121 address>`


The fixed value of decimal 254, which converts to hex FE, should always be used for MAC-based SNPA address masks.

For a full description of address formats, see the *IRIS OSI Transport Service Network Directory Compiler Guide*.

SYSTEMS

An entry is included for each known intermediate system with details of the subnet to which that intermediate system is attached, along with the SNPA address. For example:



```
achil:  {
                pdn = 1,/3259002919
                ether2 = 0,/080069021B9FFE,ESIS
        }
```

The logical network port is used by the network layer to route packets to one subnet or another. The number is chosen as follows:

- Logical Port 0 is **first** LAN card
- Logical Port 1 is **next** LAN card
  •
  •
  •
- Logical Port N is **last** LAN card
- Logical Port N + 1 is **SNDCF protocol** (route to all WAN cards)

The logical port should not be confused with the physical WAN card port numbering system, which is specified using the **`dtemap`** and **`sndcmap`** utilities (see page 5-2 and page 4-28, respectively).

DIRECTORIES

An entry is included here for each intermediate system for which you want to generate a routing table. The actual file created will be:

```
<system_name>.rt
```

In the example above, three files will be created after the NDC is run:

- **`achil.rt`**
- **`aran.rt`**
- **`bofin.rt`**

C. REFERENCES

Contents

The documents listed in this appendix relate to the UNIX and STREAMS concepts used in this manual.

C.1. UNIX and STREAMS References

1. *AT&T UNIX System V/386 Network Programmer's Guide*, Prentice Hall, Englewood Cliffs, New Jersey 07632, 1988. (AT&T Select Code 307–013)
2. *AT&T UNIX System V/386 STREAMS Primer*, Prentice Hall, Englewood Cliffs, New Jersey 07632, 1988. (AT&T Select Code 307–229)
3. *AT&T UNIX System V/386 STREAMS Programmer's Guide*, Prentice Hall, Englewood Cliffs, New Jersey 07632, 1987. (AT&T Select Code 307–227)
4. *AT&T System V Interface Definition, Volume 3*. (AT&T Select Code 307–227)

C.2. General ISO/OSI References

1. ISO 7498, Information Processing Systems — Open Systems Interconnection — *Basic Reference Model*, October 1984.
2. NBS–SP 500–162, *Stable Implementation Agreements for OSI Protocols*, Version 2, Edition 1, December 1988.
3. TOP 3.0, *Technical and Office Protocol Specification*, Version 3.0.
4. GOSIP, *U.S. Government Open Systems Interconnection Profile*, Version 1, June 1988.

C.3. Transport Layer References

1. ISO 8072, Information Processing Systems — Open Systems Interconnection — *Transport Service Definition*, June 1986.
2. ISO 8073, Information Processing Systems — Open Systems Interconnection — *Transport Protocol Specification*, July 1986.
3. ISO 8072/AD1, *Addendum 1 to the Transport Service Definition Covering Connectionless–mode Transmission*, July 1986.
4. ISO 8073/DAD2, *Addendum 2 to the Transport Protocol Definition Covering Class Four Operation over Connectionless Network Service*, July 1987.
5. ISO 8602, Information Processing Systems — Data Communications — *Protocol for Providing the Connectionless–mode Transport Service*, July 1987.
6. CCITT X.224, 1988 Version, Recommendation X.224, Transport Service Definition for Open Systems Interconnection (OSI) for CCITT Applications.

C.4. Network Layer References

1. ISO 8348, Information Processing Systems — Data Communications — *Network Service Definition*, April 1987.
2. ISO 8348/AD1, Information Processing Systems — Data Communications — Network Service Definition — *Addendum 1: Connectionless–mode Transmission,* April 1987.
3. ISO 8348/AD2, Information Processing Systems — Data Communications — *Addendum to the Network Service Definition Covering Network Layer Addressing,* March 1988.
4. ISO 8648, Information Processing Systems — Data Communications — *Internal Organization of the Network Layer*, February 1988.
5. ISO 8208, Information Processing Systems — Data Communications — *X.25 Packet Level Protocol*, September 1987.
6. ISO 8878, Information Processing Systems — Data Communications — *Use of X.25 to Provide the Connection–mode Network Service*, September 1987.
7. CCITT Red Book, Volume VIII, Fascicle VIII.3, Recommendation X.25, *Interface Between DTE and DCE for Terminals Operating in the Packet Mode and Connected to Public Data Networks by Dedicated Circuit.*
8. ISO 8473, Information Processing Systems — Data Communications — *Protocol for Providing the Connectionless–mode Network Service,* January 1988.

C.5. LAPB Data Link Layer References

1. ISO 7776, Information Processing Systems — Data Communications — 2nd DP 7776 Revised — *Description of the 1984 X.25 LAPB — Compatible DTE Data Link Procedures*, December 1986.
2. ISO/DIS 8886.3, Information Processing Systems — Data Communications — *Data Link Service Definition for Open Systems Interconnection*, June 1988.

GLOSSARY

This glossary defines key terms and acronyms that appear in this manual. Italicized terms within definitions are defined elsewhere in this glossary.

call – An X.25 term that is analogous to initiating a connection request in general OSI terminology.

CLNS – connectionless network service. A network layer service that uses *connectionless* mode (also called datagram mode) to provide communications services to *NSUs*. *CLNP* is an example of a CLNS provider.

CLNP – Connectionless Network Protocol, defined in IS0 8473. The protocol governing the basic connectionles-mode service that LANs can provide.

confirm or **confirmation** – An event that usually results when a *peer entity* issues a *response*, such as the response to a connection *request*.

connection–oriented – A mode of communication between two *peer entities* in which a logical connection between the two entities is established before data is transferred.

connectionless – A mode of communication in which an *entity* may transmit a message without first establishing a connection with another entity. The message may be directed to one or more entities on the network.

CONS – Connection–Oriented Network Service. A Network layer service that uses *connection–oriented* mode to provide communications services to *NSUs*. *X.25 PLP* is an example of a CONS provider.

DCE – Data Communications Equipment. One of two systems that comprise an X.25 *WAN* connection (the other is *DTE*). It is the packet switch within an X.25 subnetwork that is responsible for conveying data received from one *DTE* to another.

downstream – Refers to the direction from a STREAM head towards a STREAMS driver.

driver – An *entity* in a STREAMS protocol stack. Every STREAMS protocol stack has one device driver, which is located at the Stream's end and interfaces to hardware. The STREAMS protocol stack may also have one or more pseudo–device drivers (or multiplexing drivers).

DTE – Data Terminal Equipment. One of the two systems that comprise an X.25 *WAN* connection (the other is *DCE*). It is an end system that establishes and terminates communications.

endpoint – The point of communication between a service user and its service provider. Each user of a provider has its own endpoint, which it uses to establish an identity with the provider. Typically, an address is associated with an endpoint.

entity – An addressable unit of software or hardware that provides a service to or makes use of a service provided by another entity. An implementation of an OSI protocol within a system. An entity resides entirely within one layer.

event queue – If data can be received faster than an *entity* can process it, the data is placed into an event queue. The queue maintains the sequence of reception, and holds the data until it can be processed. Note that "event" in this case is a general term, and is not necessarily an *interface event* as described above.

host – A computer, such as a 386–based IBM model, into which communications hardware and software have been installed.

indication – An *event* that usually originates when a remote system issues a *request*. For example, when a *peer entity* issues a connection request, the entity to which the request is directed receives a connect indication.

initiator – A network station that initiates a connection with another station (a *responder*).

interface – A logical data path between two *entities* in different layers.

INLP - Inactive Network Layer Protocol.

IP – Internetwork Protocol. The name of an OSI network (layer 3) protocol that provides *connectionless* network service (*CLNS*) to either the *connection–oriented* or *connectionless* transport protocol. CLNP is also known as IP.

IPDU - Internetwork Protocol Data Unit. See also *PDU*.

LAN – local area network. A network designed to provide communications services over a relatively short distance, such as a single site.

LAPB – Link Access Procedure Balanced. A Data Link layer point–to–point communications protocol for use in Wide Area Network environments.

LLC – logical link control. The name of a data link (layer 2) protocol.

LSAP - LLC Service Access Point - See also *LLC*.

MAC entity – media access control entity. A sublayer within the data link layer that is device–dependent. The LT–610 package is designed to interface to most standard MAC entities.

message – An arbitrarily long unit of data that can be transmitted over a network (such as the contents of a file). The size of a message is not limited by the size of buffers used within the *protocol stack* used to transmit it.

NIDU – Network Interface Data Unit. The data and control information that is in transit between the *NSU* and Network layer.

NSAP – network service access point. A virtual point through which a network service user may access Network layer services.

NSAP address – The physical value that is associated with an *NSAP*.

NSDU – Network Service Data Unit. A unit of Network data contained within an *NSU* buffer. When transferred to the Transport layer, an NSDU becomes an *NIDU*.

NSU – Network Service User. A software *entity* that uses the services of the Network layer.

PDU - Protocol Data Unit. A unit of data containing protocol information of relevance to a peer layer in an OSI stack.

peer entity – The *entity* on the other system. Typically, two entities (each on a different system) communicate with each other through their respective *protocol stacks*. For example, a Transport layer entity may transfer data to the peer Transport layer entity with which it has established a connection.

PLP – Packet Level Protocol. The name of a Network (layer 3) protocol used for *connection–oriented* operation.

protocol stack – A layered organization of *entities* executing on one system that can communicate with other systems. It consists of an application entity that originates data transmission, and several underlying entities that provide the required communications services.

protocol suite – STREAMS/UNIX terminology for a *protocol stack*.

request – An action that initiates the transfer of data between systems. For example, an *initiator* may request that a connection be established.

responder – A system that responds to a connection request from an *initiator.*

response – An action that an *entity* may issue in response to an *indication*. For example, when a *peer entity* issues a connection *request*, the responding entity issues a connection response that either accepts or rejects the connection.

segmented data – A *message* that spans several buffers. The ISO specifies when and how data can be segmented.

service provider – An *entity* that provides service for the next higher entity on a *protocol stack*. For example, the Physical layer is a service provider to the Data Link layer.

service user – An *entity* that uses the service of the next lower entity in a *protocol stack*. For example, a Transport entity is the service user of a Network entity.

SNDCF – Sub Network Dependent Convergence Functions. A sublayer within the Network layer. For example, one type of SNDCF *entity* interfaces a *connectionless* entity, such as *IP*, to a *connection–oriented* entity, such as *PLP.*

SNPA Address – subnetwork point of attachment address. A standard OSI address type which consists of the address of the *MAC entity* providing the service to LLC1 and the LLC1 service user, and the LSAP selector of the LLC1 service user.

service user – An *entity* that uses the service of the next lower entity in a protocol stack. For example, a Transport entity is the service user of a Network entity.

Stream – A data flow path between a *Stream Head* and *driver* in a STREAMS protocol stack.

Stream head – An *entity* in a STREAMS protocol stack that interfaces with a user process providing the interface between the Stream in kernel space and the user application in user space. The Stream head processes STREAMS system calls from the user application, and allows data to be passed between the user application and the Stream in both directions.

STREAMS – A combination of system calls, kernel routines, and kernel utilities in UNIX System V Release 3 operating system that provide an efficient means of implementing a dynamic network stack. STREAMS defines a standard interface between a STREAMS–based user application and the *STREAMS protocol stac*k with which it communicates.

TIDU – transport interface data unit. Data and control information in transit between the TSU and Transport layer.

TLI – Transport Layer Interface. The standard interface to the Transport layer provided in the AT&T UNIX System V Release 3 operating system.

TP0/2/4 – Transport Protocol, Classes 0, 2, and 4.

TP4 – transport protocol class 4. The class of transport service that can communicate with *IP.*

TPDU – Transport Protocol Data Unit. A complete unit of Transport data and associated control information. One or more TPDUs make up a *TSDU*. A TPDU may be *segmented* into two or more *TIDUs* for transmission across the Transport/TSU *interface*.

TSAP – Transport Service Access Point. A virtual point through which a *TSU* may access Transport layer services.

TSAP selector – The relative name of a *TSAP.*

TSDU – transport service data unit. The unit of data conveyed by the transport service. When transferred to the transport layer, a TSDU may be segmented into two or more *TIDUs*, then reassembled into a *TPDU* when it reaches the transport layer.

TSU – transport service user. An OSI *software entity* that is a *service user* of the transport layer.

upstream – Refers to the direction from a STREAMS driver to the *Stream head.*

user data – Data that originates from a *service user.*

Virtual Circuit – The X.25 layer 3 protocol provides two forms of connection between a pair of DTEs: virtual circuit (VC) and permanent virtual circuit (PVC). A VC requires the three phases of a connection: connection estabablishment, data transfer, and connection release.

WAN – Wide Area Network. A network whose size and geographical area of coverage is larger than a single site.

INDEX

Tell Us About This Manual

As a user of Silicon Graphics products, you can help us to better understand your needs and to improve the quality of our documentation.

Any information that you provide will be useful. Here is a list of suggested topics:

- General impression of the document
- Omission of material that you expected to find
- Technical errors
- Relevance of the material to the job you had to do
- Quality of the printing and binding

Please send the title and part number of the document with your comments. The part number for this document is 007-1547-020.

Thank you!

Three Ways to Reach Us

- To send your comments by **electronic mail**, use either of these addresses:
  - On the Internet: techpubs@sgi.com
  - For UUCP mail (through any backbone site): *[your_site]*!sgi!techpubs
- To **fax** your comments (or annotated copies of manual pages), use this fax number: 650-932-0801
- To send your comments by **traditional mail**, use this address:

  Technical Publications
  Silicon Graphics, Inc.
  2011 North Shoreline Boulevard, M/S 535
  Mountain View, California 94043-1389