NetFX

Fibre Channel Adapters

User's Guide and Programmer's Reference

Document No. IG1000011-001
Revision 3.2
July 1999

<u>SOFTWARE LICENSE AGREEMENT</u>

<u>This software package is protected by federal copyright law and international treaty.</u>

By law, you, the end user, may do the following:

a) make a copy of the software solely for backup or archival purposes; or

b) transfer the software to a single hard disk, provided you keep the original solely for backup or archival purposes.

Federal copyright law prohibits you from making any other copy of the software for any other reason without our permission. Federal copyright law also prohibits you from copying the written materials accompanying the software without first obtaining our permission.

For a copy of the brochure, Software Use and the Law, please send a stamped, self-addressed, business size envelope to:
Software Publishers Association, 1101 Connecticut Avenue., NW, Suite 901, Washington, DC 20036.

THIS IS A LICENSE AGREEMENT AND NOT AN AGREEMENT FOR SALE. A LICENSE AGREEMENT IS A LEGAL AGREEMENT BETWEEN YOU, THE END USER, AND PRISA NETWORKS, INC. PLEASE READ THIS LICENSE AGREEMENT CAREFULLY BEFORE OPENING THIS PACKAGE. IF YOU DO NOT AGREE WITH THE TERMS AND CONDITIONS OF THIS LICENSE AGREEMENT, PROMPTLY RETURN THE PACKAGE UNOPENED AND YOUR MONEY WILL BE REFUNDED.

This software can only be used on a single terminal connected with a single computer at any one time. This means that the software should be loaded on only one hard drive at a time. If you wish to use this software on more than one computer, you must either erase the software from the first hard drive when you move the software to a second hard drive, or else purchase two copies of the software. You may not under any circumstances have this program loaded onto the hard drives of two or more computers at the same time. You also may not copy this software onto a hard drive and then use the media on another computer.

This is a single-copy version of NetFX. This software may not be loaded onto a Local Area Network (LAN). If you want to use our software on a LAN, even if you don't expect to have more than one person use it at a time, you must purchase the LAN-version.

You may not rent, lend, lease, or assign this software. You may, however, transfer the software and the accompanying written materials on a permanent basis. When you transfer the software on a permanent basis, you may not keep any copies, and you must remove the software from your hard disk. Also, the person to whom you transfer the software must agree to the terms of this License.

You may not alter, modify or adapt the software or accompanying materials. In addition, you may not translate, decompile, disassemble or reverse engineer the software. You also may not create any derivative works based on this software. A derivative work is defined as a translation or other form in which the software may be recast, transformed or adapted.

PRISA Networks warrants that this program will perform in substantial compliance with the published specifications and documentation supplied in this package, provided it is used on the computer hardware and with the operating system for which it was designed. This warranty is limited to a period of 90 days from the date of the original purchase.

If you report a significant defect in performance, in writing within 90 days of purchase, PRISA Networks will attempt to correct it or, at its option, authorize a refund of your license fee.

Except as specifically provided above, PRISA Networks makes no other warranty or representation, either express or implied, with respect to this software, the media or documentation, including their quality, merchantability, or fitness for a particular purpose.

In no event will PRISA Networks be liable for direct, indirect, special, incidental, or consequential damages arising out of the use or inability to use the software or documentation, even if PRISA Networks has been advised of the possibility of such damages.

The warranties and remedies set forth above are exclusive and in lieu of all others, oral or written, express or implied.

PRISA Networks is not responsible for any costs including, but not limited to, those incurred as a result of lost profits or revenue, loss of time or use of the software, loss of data, the costs of recovering such software or data, the cost of substitute software, claims by third parties, or other similar costs. In no case will PRISA Networks's liability exceed the amount of the license fee.

The warranties provided give you specific legal rights. You may have other rights which vary from state to state. Some states do not allow the exclusion of incidental or consequential damages, or the limitation of how long an implied warranty lasts, so some of the above may not apply to you.

Manufacturer's Regulatory Declarations

The NetFX-GIO64, NetFX-HIO64, and NetFX-PCI32 Fibre Channel adapter cards, with Prisa Networks supplied GLMs, conform to several national and international specifications and European Directives listed on the "Manufacturer's Declaration of Conformity," which is included with each product. The CE mark insignia displayed on each device is an indication of conformity to aforementioned European requirements.

A new "Manufacturer's Declaration of Conformity" may be provided with upgrade or option packages, and may contain upgrade information pertaining to your new NetFX-GIO64, NetFX-HIO64, or NetFX-PCI32 configuration.

CAUTION

Your NetFX-GIO64, NetFX-HIO64, and NetFX-PCI32 Fibre Channel adapters have several governmental and third-party approvals, licenses, and permits. Do not modify these products in any way that is not expressly approved by Prisa Networks. If you do, you may lose these approvals and your governmental agency authority to operate these devices.

CLASS A EQUIPMENT

FCC Notice

This equipment has been tested and found to comply with the limits for a Class A digital device, Pursuant to Part 15 of the FCC Rules. These limits are designed to provide a reasonable protection against harmful interference when the equipment is operated in a commercial environment. The equipment generates, uses, and can radiate radio frequency energy and, if not installed and used in accordance with the instruction manual, may cause harmful interference to radio communications. Operation of this equipment in a residential area is likely to cause harmful interference in which case the user will be required to correct the interference at his own expense.

Any changes or modifications not expressly approved by the manufacturer could void the user's authority to operate the equipment.

CAUTION

Shielded I/O cables must be used when operating this equipment when configured as a single or dual electrical Fibre Channel adapter.

Industry Canada Compliance Statement

This equipment does not exceed Class A limits per radio noise emissions for digital apparatus set out in the Radio Interference Regulation of Industry Canada. Operation in a residential area may cause unacceptable interference to radio and TV reception requiring the owner or operator to take whatever steps are necessary to correct the interference.

Avis de conformite aux normes du ministrere des Industrie Canada

Cet equipment ne depasse pas les limites de Classe A d'emission de bruits radioelectriques pour les appareils numerriques tells que perscrites par le Reglement sur le brouillage radioelectrique etabli par le ministere des Industrie Canada. L'exploitation faite en milieu residentiel peut entrainer le brouillage des receptions radio et tele, ce qui obligerait le proprietaire ou l'operateur a pendre les dispositions necessarires pour en eliminer les cause.

Compliance to European Directives 89/336/EEC and 93/68/EEC, Electromagnetic Compatibility

These devices complies to electromagnetic emissions limits of EN55022, C.I.S.P.R. Publication 22 and electromagnetic immunity to the European Generic Immunity Test Standard EN 50082-1, and to the test procedures described in Parts 2, 3, and 4, of the IEC 801 series of test specifications.

WARNING

This is a Class A product. In a domestic environment this product may cause radio interference in which case the user may be required to take adequate measures.

Laser Safety Compliance

The optical NetFX-GIO64, NetFX-HIO64, and NetFX-PCI32 use GLMs that are certified as a Class 1 laser product per 21 CFR (U.S. Code of Federal Regulations), Subchapter J. A Class 1 laser product is safe for use and does not pose a biological hazard if used in accordance within the limits and instructions.

CAUTION

There are no user serviceable parts nor any maintenance required for the optical GLMs used in the optical NetFX-GIO64, NetFX-HIO64, and NetFX-PCI32. All adjustments are made at the factory of the GLM supplier before shipment to our customer. Tampering with, modifying or breaking the preset trim pot seals will result in voided product warranty. It may also result in improper operation of the GLM circuitry, and possible overstress of the laser source. Device degradation or product failure may result.

Connection of the GLM to a non-approved optical source, operating above the recommended absolute maximum conditions or operating the GLM in a manner inconsistent with its design and function that are provided by the NetFX-GIO64, NetFX-HIO64, and NetFX-PCI32 may result in hazardous radiation exposure and may be considered an act of modifying or manufacturing a laser product. The person(s) performing such act is required by law to recertify and reidentify the laser product under the provisions of 21 CFR(Subchapter J).

Adapter Handling Instructions

- Before handling the boards for any reason, put on a properly-grounded, anti-static wrist strap to prevent the discharge of static electricity, which can damage the adapter.
- Keep boards sealed in their anti-static, protective shipping bags until you're ready to install them.
- Handle the NetFX adapter and all computer components gently. Do not force, twist or apply excessive pressure while installing.
- *Do not* remove any Prisa-supplied GLMs from NetFX-PCI32 adapters.
- See the manufacturer-supplied manual for detailed information about your machine.

Customer Service

If you experience problems that you are unable to resolve, contact the technical support representative at the company that supplied your Prisa product.

Prior to you call, it is helpful to gather the following information:

- ❒ Machine(s) Operating System level, including SGI patches
- ❒ Hardware configuration (typically an output of “hinv”)
- ❒ Prisa Networks software release level
- ❒ Other software installed on the system (e.g. INFERNO, Cineon, etc.)
- ❒ Contents of any error log files (e.g. /var/adm/SYSLOG)
- ❒ List of NetFX Diagnostic test failures
- ❒ A list of the procedures you performed prior to any error

Contents

About This Guide

Welcome to the *NetFX User's Guide*! This guide introduces you to NetFX and Fibre Channel technology. It contains step-by-step instructions that explain how to install, test, and configure Prisa's family of NetFX adapters. For detailed information about your adapter, refer to the manual pages delivered with the NetFX software.

This guide includes these chapters:

- Chapter 1, "The Prisa Solution," introduces Fibre Channel and explains how this technology sets a new network standard. It also describes the Prisa Networks complete Fibre Channel solution.
- Chapter 2, "Designing Your Fibre Channel Network," describes Fibre Channel topologies, explains some different Storage Configurations, and provides anatomy information for NetFX Adapters.
- Chapter 3, "Installing NetFX Software," contains instructions that explain how to install NetFX software.
- Chapter 4, "Installing NetFX Hardware," contains step-by-step instructions that show you how to handle, install, and remove adapters.
- Chapter 5, "Running Hardware Diagnostic Tests," explains how to run internal Pass/Fail diagnostic tests to verify the NetFX adapters are installed correctly. It also explains how to complete channel and board level tests.
- Chapter 6, "Configuring NetFX Software," explains how to configure NetFX software.
- Chapter 7, "About NetFX Software," describes how to use NetFX application, development, network, and storage software to maximize network performance.
- Appendix A, "Programmer's Guide," contains detailed information about NetFX APIs.
- Appendix B, "Loop ID Reference Table," lists valid Loop IDs available from the Fibre Channel Arbitrated Loop Direct Attach SCSI Profile (Private Loop), version 2.0.
- Appendix C, "The Fibre Channel Hierarchy", gives some detailed technical information about the Fibre Channel Protocol.

User Guide Conventions

As you use with this guide, follow these conventions:

Convention	Meaning
`Courier`	The Courier font identifies information you need to type exactly as shown and manual pages.
Italic	The italic style identifies information provided for reference purposes.
Bold Black	Bold black text identifies required attribute settings.
Bold Gray	Bold gray text identifies optional attribute settings.
`< >` Angle Brackets	Angle brackets indicate where to type information specific for your site, such as the workstation's name.
🛑 Stop	A stop identifies important information you need to know to ensure you don't inadvertently perform an invalid operation that could damage your adapter.
⚠ Note	A note identifies information useful to know about a command, attribute, or procedure.
⚡ Warning	A warning identifies tasks that affect other procedures or required value.

Terms and Acronyms

This section lists and describes terms and acronyms found in this document, complementary documentation, and NetFX software.

Terms

NLPorts	NLPorts are attached using an arbitrated loop network configuration.
NLPort vs NPort	When you work with NetFX software, the term NLPort means the same thing as NPort in the programs and corresponding documentation. This means the software, which manages NPort and NLPort required values, refers to both NPorts and NLPorts.
NPorts	NPorts are attached using a point-to-point or switch network

configuration.

Acronyms

AC	Alternating Current
ANSI	American National Standards Institute
API	Application Programming Interface
ATM	Asynchronous Transfer Mode
BDS	Block Data Service
DMA	Direct Memory Access
EFS	Error Free Seconds
EISA	Extended Industry Standard Architecture
ESD	Electro Static Discharge
FC-AL	Fibre Channel Arbitrated Loop
FCCP	Fibre Channel Copy Program
FCP	Fiber Channel Protocol
FCSI	Input/Output File System
FDDI	Fiber Distributed Data Interface
FIFO	First In, First Out
FSD	Fibre Channel SCSI Driver
Gb	Gigabit
GB	Gigabyte
GLM	Gigabaud Link Module
GUI	Graphical User Interface
HIPPI	High Performance Parallel Interface
I/O	Input/Output
IP	Internet Protocol
IRIX	SGI Operating System
JBOD	Just a Bunch of Disks
LAN	Local Area Network
LED	Light Emitting Diode
LUN	Logical Unit Number

NFS	Network File System
OFC	Open Fiber Control
RAID	Redundant Array of Inexpensive Disks
RPC	Remote Call Procedure
RGB	Red Green Blue (a color model)
SCSI	Small Computer System Interface
SDI AP	Serial Data Interface Application Programming Interface
SGI	Silicon Graphics Inc.
TCP	Transmission Control Protocol
TCP/IP	Transmission Control Protocol/Internet Protocol
ULP	Upper Level Protocol
URL	Uniform Resource Locators
WWN	World Wide Name

1 The Prisa Solution

Overview

This chapter describes the Prisa NetFX Fibre Channel solution. We introduce Fibre Channel and explain why and how this technology sets a new network standard. We also provide detailed information about the complete Prisa NetFX Fibre Channel solution. If you're already familiar with the power and flexibility of Fibre Channel and the Prisa NetFX product family, you can skip to *Chapter 2 Designing a Fibre Channel Network.*

This chapter includes these sections:

Fibre Channel Performance

The growth in Local Area Network (LAN) computing, coupled with improved workstation and server performance, defines a new business requirement: a network with a data transfer rate of one gigabit per second. Network standards, as *Figure 1-1 History of Network Bandwidth Demand and Growth* shows, have evolved with industry's escalating need to move data. Within a 15-year time period, bandwidth has undergone dramatic growth. Today, Fibre Channel provides the ideal interconnect solution.

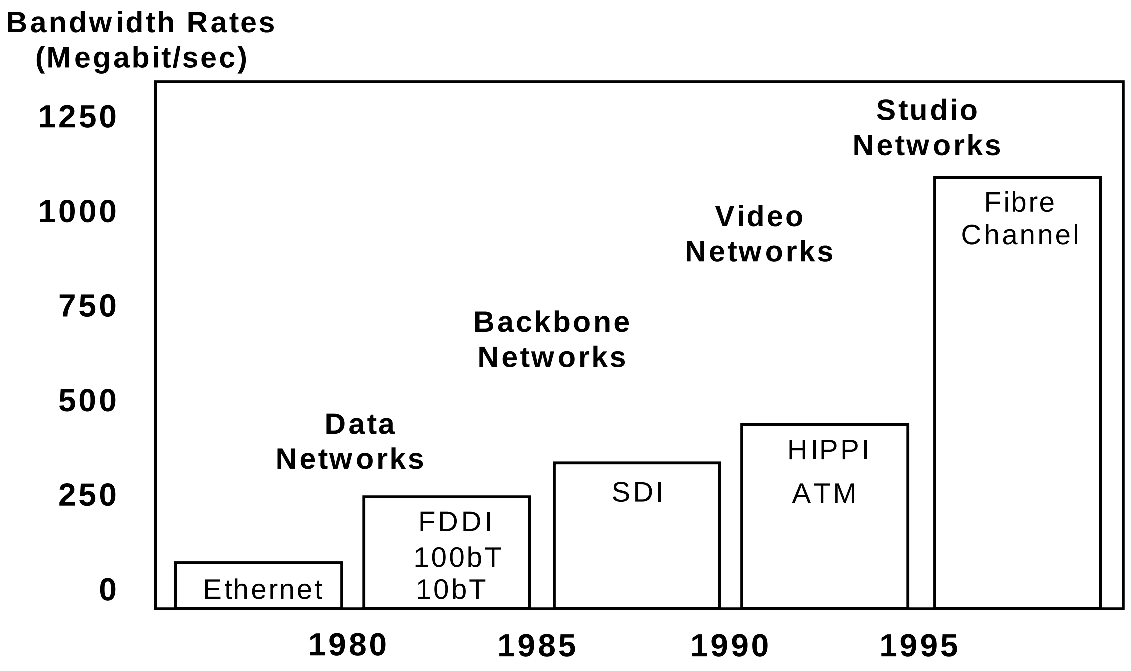


Figure 1-1 History of Network Bandwidth Demand and Growth

How Fibre Channel Works

Fibre Channel is a family of ANSI standards designed specifically to provide high-speed data transfer between workstations, personal and large computers, disk drives, peripherals, and display devices. It transfers data between a buffer at the source device (such as a RAID drive) and another buffer at the destination device (such as a workstation). Fibre Channel ignores the data itself and how it's formatted—it simply takes what is in the sending buffer and transports it to the receiving buffer as quickly as possible.

Fibre Channel:

- Accommodates multiple protocols simultaneously, such as SCSI, TCP/IP, HIPPI, ATM, Ethernet, and IPI. This makes Fibre Channel the best choice for environments using a variety of computing and peripheral equipment.
- Combines the best attributes of a channel (such as handling simple error correction in hardware) with those of a network (such as passing more complex error recovery to the central processor).
- Supports point-to-point, arbitrated loop, and switched topologies. This flexibility allows you to choose the configuration that best meets your cost and performance requirements.

Applying Fibre Channel Technology

Fibre Channel allows organizations to focus on their business instead of continually responding to data transfer bottlenecks. Here are only some of the industries that would benefit from Fibre Channel technology:

- Film industry post production that perform digital special effects and digital film editing.
- Hospital and medical centers that transmit x-ray and magnetic resonance scan images.
- Universities and government laboratories performing scientific visualization associated with atmospheric and energy research.
- Corporate and government research centers involved with complex, computer-aided design activities.
- Service organizations, such as the Post Office, Internal Revenue Service, and insurance companies, that process an enormous number of images.
- Large transaction processing entities, such as banks and brokerages, with strict record-keeping and daily archiving responsibilities.

Fibre Channel Topologies

Prisa supports three network topologies: point-to-point, arbitrated loop, and switch.

Point-to-point

The point-to-point topology connects two, and only two, devices. While this topology limits the number of connected ports, it provides the simplest, most cost-effective way to move large quantities of data at the highest possible speed. The entire bandwidth is available for the transfer of data between the two connected devices.

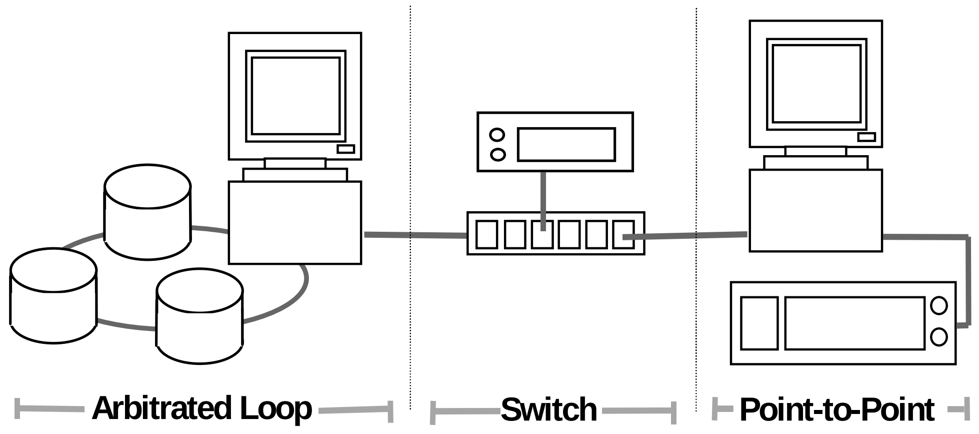
Arbitrated Loop

The arbitrated loop topology links up to 127 devices. In this configuration, the devices share (arbitrate) the bandwidth of the loop. Loops provide a cost-effective way to network multiple devices where transfers of large files are occasional and response time is critical.

Switch

In the switch topology each device is connected to the switch directly. This topology supports full bandwidth per pair of nodes.

Figure 1-2 Topologies NetFX Supports illustrates the three topologies supported and shows how a Fibre Channel network can incorporate multiple topologies in one configuration. The power and flexibility these topologies offer allows for the design of cost-effective, gigabit-per-second networks.



Prisa NetFX Technology Provides the Full Solution

Prisa's NetFX Fibre Channel meets the film industry's need to move huge video and audio data files across networks. With data transfer rates 10 to 250 times faster than other technologies, Fibre Channel solves the bottleneck problem eloquently. With NetFX you can load and transfer video images in seconds instead of minutes!

The Prisa line of NetFX adapters provide Fibre Channel connectivity to the SGI systems listed below.

GIO64 and HIO64 Buses		PCI32 Bus	
Indigo2™	Onyx®	OCTANE™	O2™
Indigo2 IMPACT™	Challenge®DM/L/XL	Origin200™	Onyx2®
		Origin2000™	

Powerful NetFX Features

NetFX includes these features:

- Point-to-point, arbitrated loop, and switch topologies supported.
- Disk and network transfer rates up to 100 megabytes per second.
- Plug-and-play interface with SGI high-speed 64-bit GIO, HIO, and PCI buses.

- Very high sustainable data throughput for large block transfers, as well as high transaction rates for small-message traffic (such as TCP/IP).
- Support for configurations that include up to two channels per adapter.
- Ability to select either optical or electrical media interface using on-board gigabaud link modules (GLM).
- A diagnostic utility that simplifies installation, start-up, and maintenance.

Hardware Overview

A NetFX adapter board plugs into a GIO, HIO, or PCI expansion slot in your SGI workstation and occupies a single slot. The adapters are available in single or dual channel configurations. The single channel NetFX adapter provides a cost-effective way to connect a single system to a single network. The two or dual channel NetFX adapter provides an effective way to connect a single system to multiple networks for increased connectivity and bandwidth. Multiple adapters can be installed in a single system (that is, a high-end server) to provide scaleable connectivity and bandwidth.

NetFX adapters can communicate via various fiber optic or copper Fibre Channel media, using standard gigabaud link module (GLM) serial communications devices. The GLM socket on the NetFX adapter allows insertion of different modules, depending on the specific combination of media desired. For dual channel NetFX adapters, each channel can use a different GLM supporting combinations of media types.

Prisa Networks offers the following GLMs as standard:

- short wave laser, OFC and non-OFC
- twin-axial copper (ECL)

⚠ OFC and non-OFC cannot operate directly together.

Software Overview

The Prisa Family of NetFX software programs provide the solution to gigabit networking requirements using SGI workstations and the NetFX adapter. NetFX software uses the latest Fibre Channel technology to provide faster-than-real-time data transfers. A brief description of the software packages available from Prisa for NetFX adapters follows. For detailed information, refer *to Chapter 7 About NetFX Software*.

NetFX Base

The base level hardware drivers and diagnostic utility for the NetFX adapter supports point to point and arbitrated loop topologies.

NetFX Switch

Enhancements to the NetFX Fibre Channel Base software package to add support for switched topologies.

NetFX SCSI Disk Driver and NetFX Fibre Channel SCSI Protocol

These two software packages implement SCSI's FCP standard enabling communication to any FCP-compliant SCSI target device (such as RAID and JBOD).

XLV Generic Disk Adaptation Driver (xlvgdisk)

Allows the NetFX SCSI Disk Driver package to be used with XLV (IRIX versions 6.3 and below).

NetFX Fibre Channel IP

Implements FCIP profile compliant IP over Fibre Channel, which allows IP interoperability with any FCIP profile compliant device.

NetFX IP Accelerator (ZIP)

Implements IP hardware acceleration for the NetFX Adapter while maintaining IP profile compliance.

NetFX Transporter

Implements a Prisa proprietary protocol for transferring large amounts of data with minimal CPU overhead. It includes the FCCP utility for copying files using the Transporter protocol.

NetFX Development Environment

Provides a programer's interface (API) to the Transporter protocol.

Prisa NetFX Enhances Performance

Prisa NetFX software and hardware are optimized to sustain high data throughput for large block size data transfers while supporting high transaction rates for small message traffic.

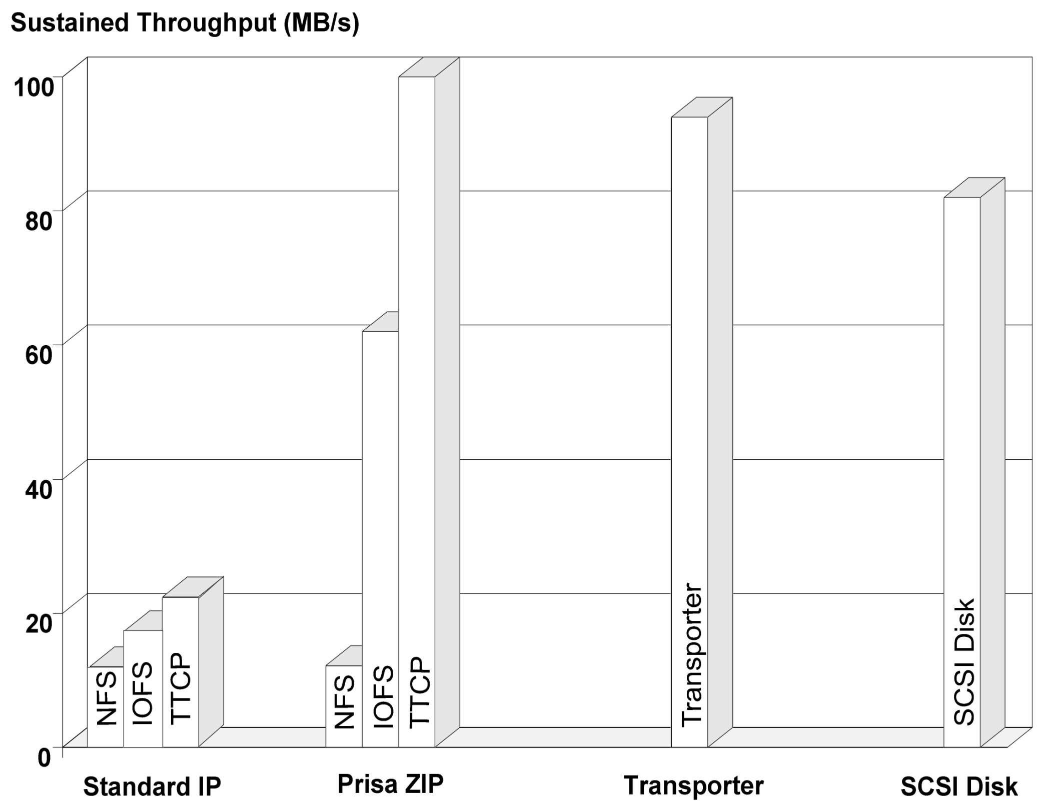


Figure 1-3 Prisa NetFX Enhances Performance

Other Prisa Networks Product Offerings

The range of Prisa NetFX products include the Prisa NetFX Loop Hub and the Prisa NetFX Switch. Combined with the single and dual channel adapters, the power and flexibility Fibre Channel offers is at your fingertips.

User's Guide Overview

This guide is divided into chapters that explain the different steps required to install and configure NetFX adapters and software. The configuration process requires these general steps:

1. Design a Fibre Channel Network, see Chapter 2.
2. Install NetFX software, see Chapter 3.
3. Install the adapter, see Chapter 4.
4. Run diagnostic tests, see Chapter 5.
5. Configure the adapter, see Chapter 6.

2 Designing Your Fibre Channel Network

NetFX at a Glance

1. Design a Fibre Channel Network.
2. Install NetFX software, see Chapter 3.
3. Install the adapter, see Chapter 4.
4. Run diagnostic tests, see Chapter 5.
5. Configure NetFX software, see Chapter 6.

Overview

This chapter describes Fibre Channel topologies and explains where to find information required to install and configure NetFX adapters.

This chapter includes these sections:

Choosing a Topology

Fibre Channel is flexible and performs in diverse configurations. To configure an adapter you need to install the board into a host machine and identify the manner in which the host and remotely connected Fibre Channel devices communicate. The manner in which the host and devices are set up is called a topology. NetFX adapters support three topologies: **Point-to-Point**, **Arbitrated Loop**, and **Switch**.

Point-to-point

The point-to-point topology connects two, and only two, devices. This topology limits the number of connected ports but provides the simplest, most cost-effective way to move large quantities of data at high speeds between two devices.

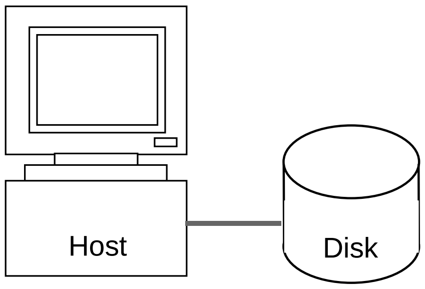


Figure 2-1 The Point-to-point Topology

Arbitrated loop

The arbitrated loop topology connects up to 127 ports that share bandwidth. In this configuration, only two ports are active at a given time. A loop topology provides a cost-effective way to network multiple devices. Even though it doesn't physically look like a loop as *Figure 2-2 The Arbitrated Loop Topology* shows, the loop hub actually forms a loop. *Figure 2-3 The Internal Design of an Arbitrated Loop* illustrates the internal design of a loop hub.

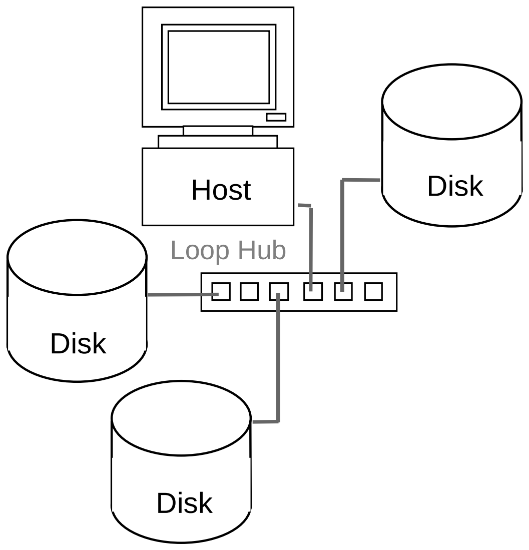


Figure 2-2 The Arbitrated Loop Topology

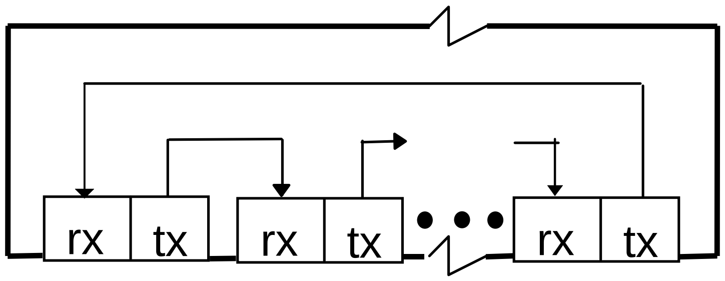


Figure 2-3 The Internal Design of an Arbitrated Loop

Switch

The switch topology connects hundreds of devices. This topology, *illustrated in Figure 2-4 The Switch Topology*, supports full bandwidth per pair of nodes simultaneously and provides a complete interconnect solution.

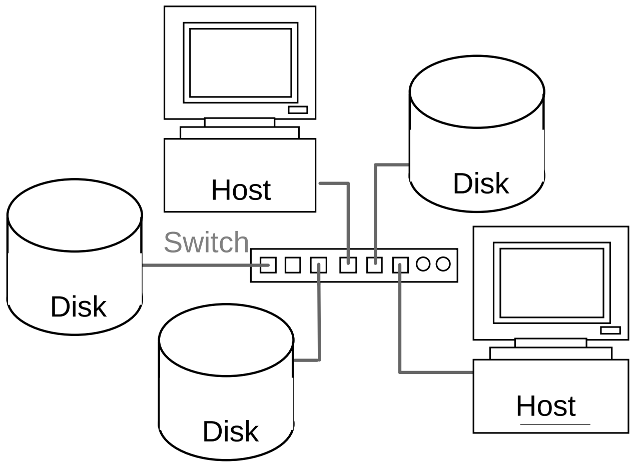


Figure 2-4 The Switch Topology

Choosing Storage Configurations

This section describes JBODs, RAIDs, and ways to achieve maximum disk throughput.

JBODs

A JBOD (Just a Bunch of Disks) has its own internal arbitrated loop. Since the bandwidth of an arbitrated loop can sustain 100 MB/sec, it is common to use software such as XLV for SGI systems to "stripe" multiple drives and have them appear as a single disk drive. Each disk inside the JBOD acts as a single NLPort. That is to say, a JBOD contains multiple NLPort devices attached to an internal arbitrated loop.

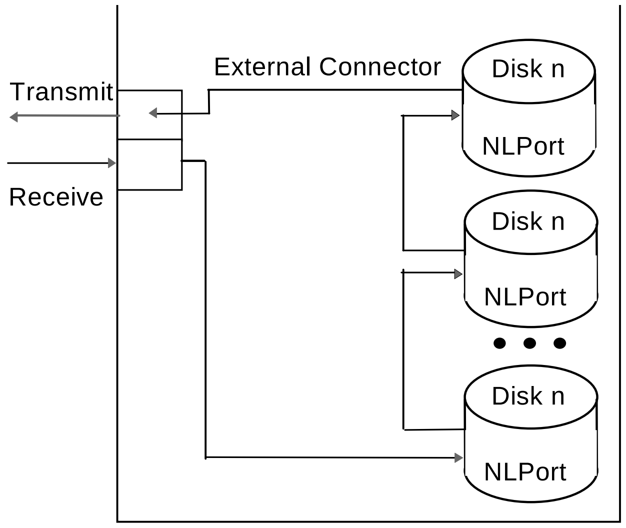


Figure 2-5 JBOD Internal Design

RAIDs

A RAID (Redundant Array of Inexpensive Disks) combines disks and striping software in one unit. These devices appear as a single device and can be attached to any topology. They are capable of sustaining up to 100 MB/sec of throughput and can be configured with many Terabytes of disk storage.

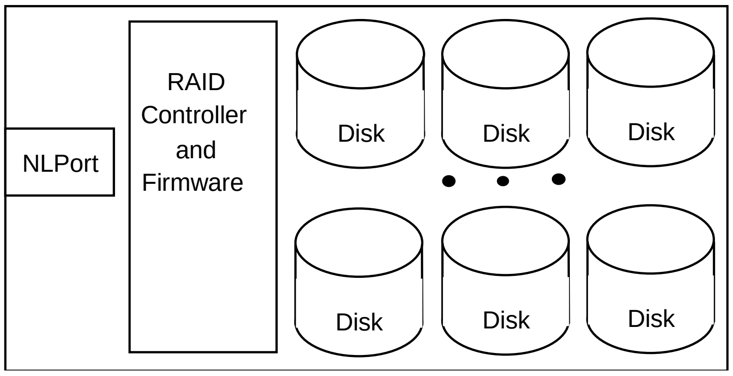


Figure 2-6 RAID Internal Design

The RAID firmware is similar to the XLV software mentioned above, but it resides on the RAID controller rather than on the host system. The RAID appears as one disk device to the host system.

Striping (XLV) across one channel

XLV software from SGI gives a JBOD RAID-like capabilities. It allows for "striping" multiple disks together into one logical drive for increased throughput and capacity. It supports disk mirroring and many more RAID-like configurations. For more information, see the SGI Insight document *IRIX ADMIN: Disks and File Systems.*

A JBOD with eight Fibre channel disk drives capable of 10MB/sec throughput and each can be striped for an effective throughput of 80 MB/sec. Similarly, if you have two RAIDs that can sustain 50 MB/sec throughput each, you can stripe across both these drives to obtain a single logical disk capable of sustaining 100 MB/sec of throughput.

Striping across multiple channels

Striping is not limited to devices attached to a single channel. You can stripe across multiple channels to increase throughput performance. For example, if you have four channels attached to a system capable of driving each channel at 100 MB/sec and four 100 MB/sec RAIDs, one attached to each channel, you could stripe the four RAIDs to achieve 400 MB/sec of disk throughput.

See *Chapter 6* for information on using logical volumes with NetFX Software.

Adapter Anatomy

Before installing your adapter, check it for the information below. This information can be useful when configuring your adapter as well as when you require technical support (e.g. you will need to have the serial number for each of your adapters for warranty support).

The NetFX-GIO adapter

The diagram shows the location of the information for GIO-based adapters.

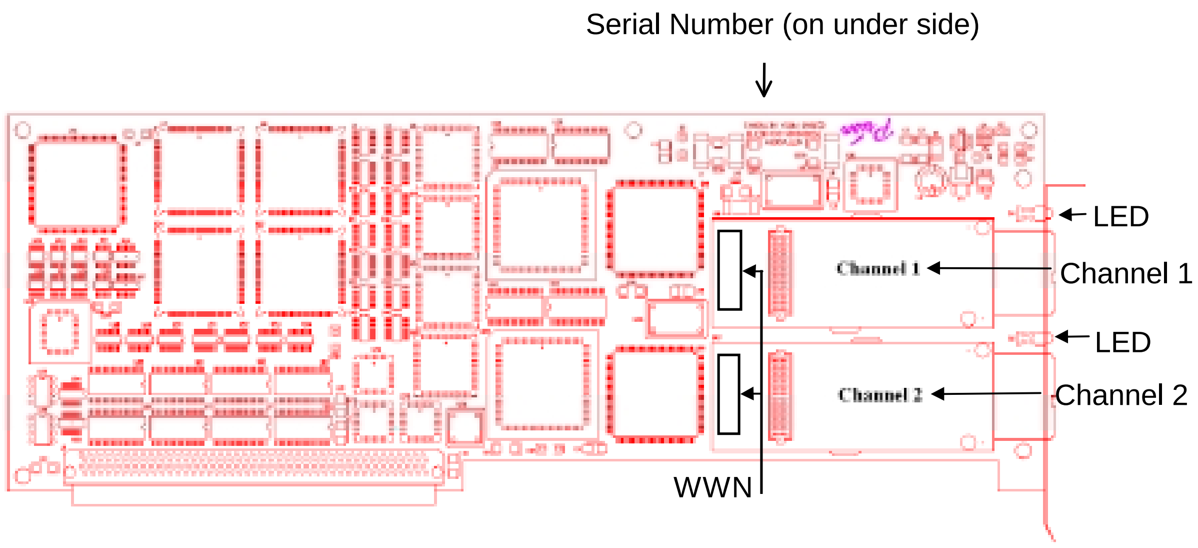


Figure 2-7 The NetFX-GIO Adapter Layout

Figure 2-9 shows the EISA/GIO backplane. SGI Indigo2 systems have three slots (four slots on the Indigo2 IMPACT). However, since the top two GIO slots (and the bottom two GIO slots on the Indigo2 IMPACT) are wired together electrically, and NetFX adapters do not support slot sharing, you can only plug in two cards at a time (one other card along with your NetFX adapter). Typically, SGI systems have a graphics card installed in slot zero.

If your system has a graphics card in slot zero, it's easiest to install your NetFX adapter in slot one. However, if an Extreme™ graphics card (which connects to one slot but covers three slots) is installed in slot zero, you'll have to move it up one slot and install your NetFX adapter in slot zero.

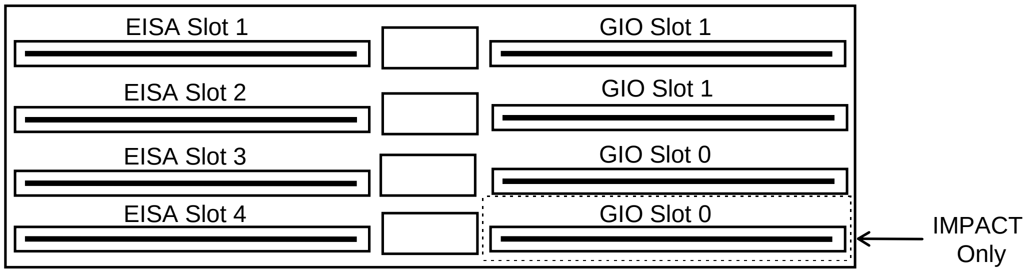


Figure 2-8 Figure 2-9 The EISA/GIO Backplane

The NetFX-HIO adapter

The diagram below shows the location of information for HIO adapters.

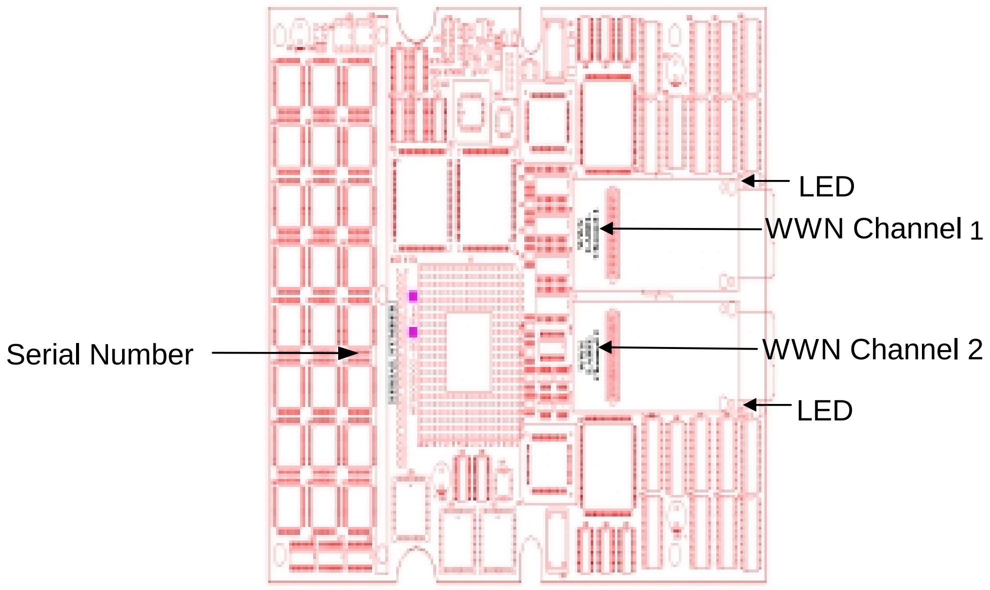


Figure 2-10 The NetFX-HIO Adapter

NetFX-PCI adapter

The diagram below shows the location of information for PCI adapters.

The slot numbering for PCI adapter varies widely from system to system. Refer to the relevant hardware manual that was shipped with your system for slot numbering information.

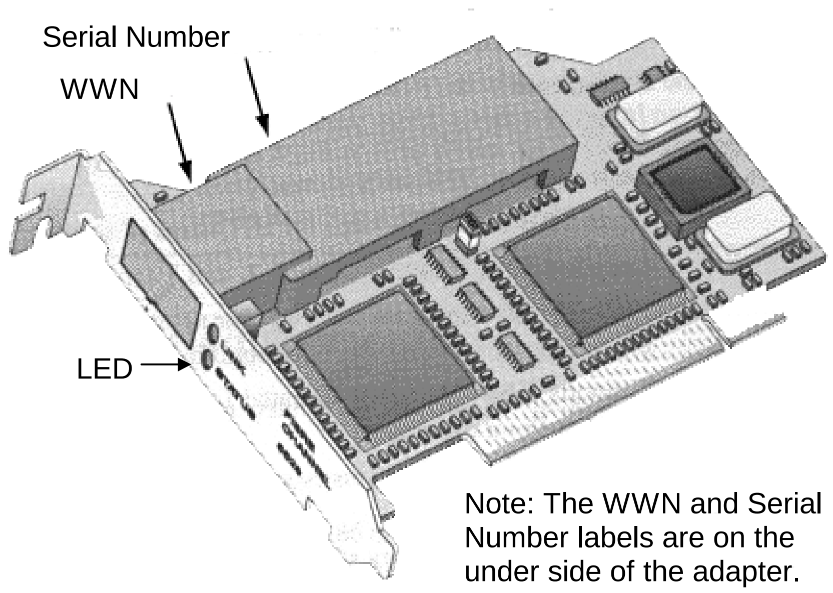

Serial Number
WWN
LED
Note: The WWN and Serial Number labels are on the under side of the adapter.

Figure 2-11 The NetFX-PCI Adapter

This page intentionally left blank.

3 Installing NetFX Software

NetFX at a Glance

1. Design a Fibre Channel Network.
2. **Install NetFX software.**
3. Install the adapter, see Chapter 4.
4. Run diagnostic tests, see Chapter 5.
5. Configure NetFX software, see Chapter 6.

Overview

This chapter explains how to install NetFX software, which includes application, development, networking, and device driver programs. For detailed information about NetFX software, see *Chapter 7 About NetFX Software*.

This chapter includes these sections:

Installing the NetFX Software

Most customers install the default NetFX software, which is packaged to meet the requirements of the majority of sites. A custom install is recommended for programmers writing application software with special requirements. `inst` is the installation tool used to install, upgrade, or remove Silicon Graphics software. To learn more about `inst`, see the man page `inst(1M)`. For the custom install, use the `keep` and `install` commands. For information about these commands, see the manual page `inst(1M)` or start the `inst(1M)` program and type the following at the `inst` prompt:

```
inst> help keep
inst> help install
```

The first two procedures, *To extract NetFX software from tape* and *To access the Release Notes*, are part of the same process.

Extract NetFX software from tape

You can not use `inst` to install software directly from the tape. You need to first extract the software from the tape, then place it in a local directory. For this and other procedures, # represents your system’s prompt; do not type #.

1. Login as **`root.`**
2. Make sure you have enough disk space on your system to extract the software.

   ```
   # df -k
   ```

   The number listed in the “avail” column shows how much space is available. For example, this information indicates approximately 985 megabytes are available. Your system should have at least 20 megabytes free, but 50 megabytes free is recommended:

Filesystem	Type	kbytes	use	avail	% use	Mounted on
/dev/root	xfs	2002730	1017711	985019	51%	/

3. Change to the temporary user directory.

   ```
   # cd /usr/tmp
   ```

4. Make a new NetFX directory.

   ```
   # mkdir netfx
   ```

5. Change to the new NetFX directory.

   ```
   # cd netfx
   ```

6. Extract NetFX software from a tape drive attached to the machine directly.

   ```
   # tar xvf /dev/tape
   ```

   -or-

   Access a tape drive over the network.

   ```
   # tar xvf guest@<remote machine>:/dev/tape
   . . .
   x 53/images/NetFX.idb, 10465 bytes, 21 blocks
   x 53/relnotes/NetFX/TC, 139 bytes, 1 block
   x 62/images/NetFX, 796 bytes, 2 blocks
   x 62/images/NetFX.idb, 16886 bytes, 33 blocks
   ```

```
. . .
```

Access the Release Notes

These steps are a continuation from the procedure above. The Release Notes contain up-to-date information about this release. Be sure to review these materials before you continue.

7. Display the version number for the operating system your workstation is running.

```
# uname -r
  6.2
```

Note the two digits in the version number because you need to use them in steps below. In this example, the site is running version 6.2 of the operating system. Your site may run a different version than the example. When you type the version number, do not include the period.

8. Access the Release Notes.

```
# 62/NetFX_grelnotes
```

Install NetFX software

For this procedure, replace <nn> with the operating system version number collected from step 7 of the procedure *To access the Release Notes and README pages* above. To complete a custom install, refer to the operating system documentation and the manual page `inst(1M)`. If you are logged in remotely, after you shutdown the system at the end of this procedure *do not* turn the workstation off.

1. Specify the location of the distribution.

```
# inst -f <nn>/images
inst>
```

2. Install the default Prisa software.

```
Inst> go
Reading installation history .. 100% Done.
Reading distribution .. 100% Done.
Checking dependencies .. 100% Done.
Installing/removing software ..   0%
Installing new versions of selected NetFX.sw
subsystems
Installing/removing software .. 12%
. . .
```

```
 Checking dependencies .. 100% Done.
 Installations and removals were successful.

 You must reboot your system to complete the
 installation.

You may continue with installations or quit
now.
```

3. Exit `inst`.

```
inst> quit
Building dynamic ELF inventory file for rqs(1)
processing .. 100% done
Invoking rqs(1) on necessary dynamic ELF
objects .. 100% Done.
```

4. Reconfigure your system’s kernel to include newly installed drivers.

```
# autoconfig -f
ld: WARNING 15: multiply defined;
lvgdisk_translate in xlvgdisk.o and
xlvgdiskstubs.o (2nd definition ignored)
```

Don’t worry if one or more messages similar to the one above appears.

5. Shutdown the machine.

```
# shutdown -y -g0
```

The NetFX software is installed.

Viewing Installation Logs for System Software

To review the results of the installation process at a later time, open the system software installation logs. For more information, see the manual page `inst(1M)`.

™ View system software installation logs.

```
# vi /var/inst/INSTLOG
```

Where to go from here

™ Continue to Chapter 4 Installing NetFX Hardware.

4 Installing NetFX Hardware

NetFX at a Glance

1. Design a Fibre Channel Network.
2. Install NetFX software.
3. **Install the adapter.**
4. Run diagnostic tests, see Chapter 5.
5. Configure NetFX software, see Chapter 6.

Overview

This chapter includes illustrations and step-by-step instructions that explain how to install and remove adapters. Before you begin, complete the procedures in *Chapter 3 Installing NetFX Software*. In addition, review the section *Guidelines* below*;* it contains warranty information and explains how to handle adapters properly.

This chapter includes these sections:

Guidelines

STOP

The Prisa warranty *does not* cover electrostatic discharge (ESD) damage, or damage related to mishandling of the adapter or components. To avoid damaging your NetFX adapter or other boards already installed in your computer, follow the guidelines below.

- Do not share slots; NetFX adapter boards *do not* support slot sharing.
- Before handling the boards for any reason, put on a properly-grounded, anti-static wrist strap to prevent the discharge of static electricity, which can damage the adapter.
- Keep boards sealed in their anti-static, protective shipping bags until you're ready to install them.
- Handle the NetFX adapter and all computer components gently. Do not force, twist or apply excessive pressure while installing.
- *Do not* remove any Prisa-supplied GLMs from NetFX-PCI32 adapters.
- Work slowly as you install and remove hardware to ensure you don't bend or crack the plastic tabs that fit into the metal case.
- See the manufacturer-supplied manual for detailed information about your machine.

Removing Installed Graphics Boards

Open the cover to gain access to the slot where you want to install an adapter.

Remove an installed graphics board

1. Put on a properly-grounded, anti-static wrist strap.
2. Position yourself so you face the back of the machine.

3. Disconnect any cables attached to the adapter (check the back of the machine).
4. Unscrew the screw fastening the slot to the bracket of the graphics board.
5. Gently pull the board to disconnect it, then pull it carefully out of the slot.

Where to go from here

➢ *Continue to one of these procedures:*

Install the NetFX-GIO64 adapter (see page 4-8).

Install the NetFX-HIO64 adapter (see page 4-14).

Install the NetFX-PCI32 adapter (see page 4-18).

Installing the NetFX-GIO64 Adapter-GIO adapter

Shut down the system

1. Login as **`root.`**
2. Shutdown the system.

```
# shutdown -y -gØ
```

3. If the system is turned on, press the power button to turn it off.

Press the button one time, quickly. If pressed too long, the machine shuts off and then turns back on. It takes a minute for the machine to shut down.

4. Disconnect the power and channel cables from the back of the machine.

Remove the casing cover

On the back of the computer, there are four plastic tabs that fit into slots in the metal case. Be careful to ensure you don't break any tabs.

1. Open the hinged front faceplate door and pull the top half down (see Figure 4-12).

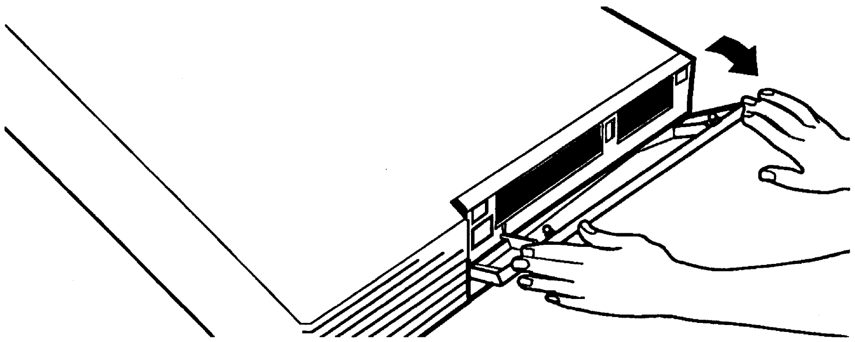


Figure 4-12 Open the Hinged Faceplate Door

2. Remove metal security bar (if installed):

   a. Unlock and remove padlock at back of computer.

   b. Slide security bar out through the front of computer (see Figure 4-13).

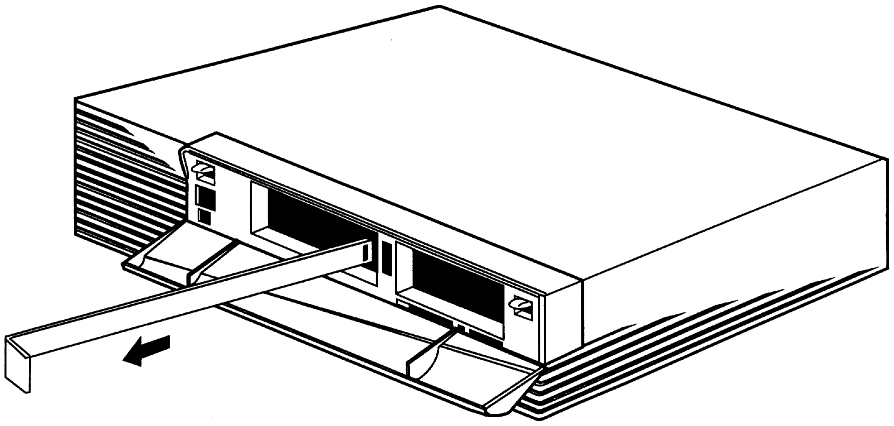


Figure 4-13 Remove the Security Bar

3. While pushing both tabs in the upper corners of the front faceplate down, tilt the entire faceplate forward about an inch and then remove it from cover (see Figure 4-14).

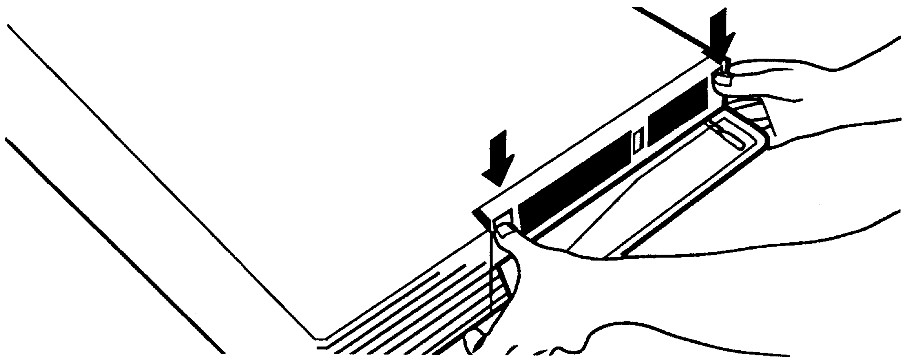


Figure 4-14 Remove the Faceplate

4. Being careful not to bend or crack the four plastic tabs that fit into slots in the metal case, lift both tabs in the lower corners on the front of the cover. Next, raise the cover just enough to clear the interior metal casing (see Figure 4-15). Push the cover back slightly to disengage four plastic tabs from the metal slots, then rotate the cover back and away from the chassis and remove it completely.

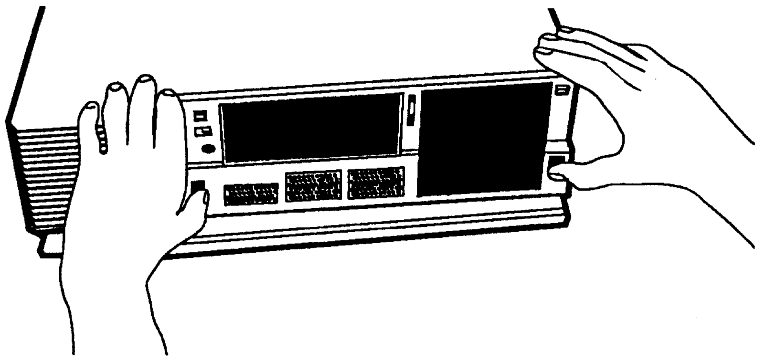


Figure 4-15 Remove the Cover

5. To open the hinged I/O cage door (on the side of the computer), gently pull the door up and back. Next, lower the door to the work table (see Figure 4-16).

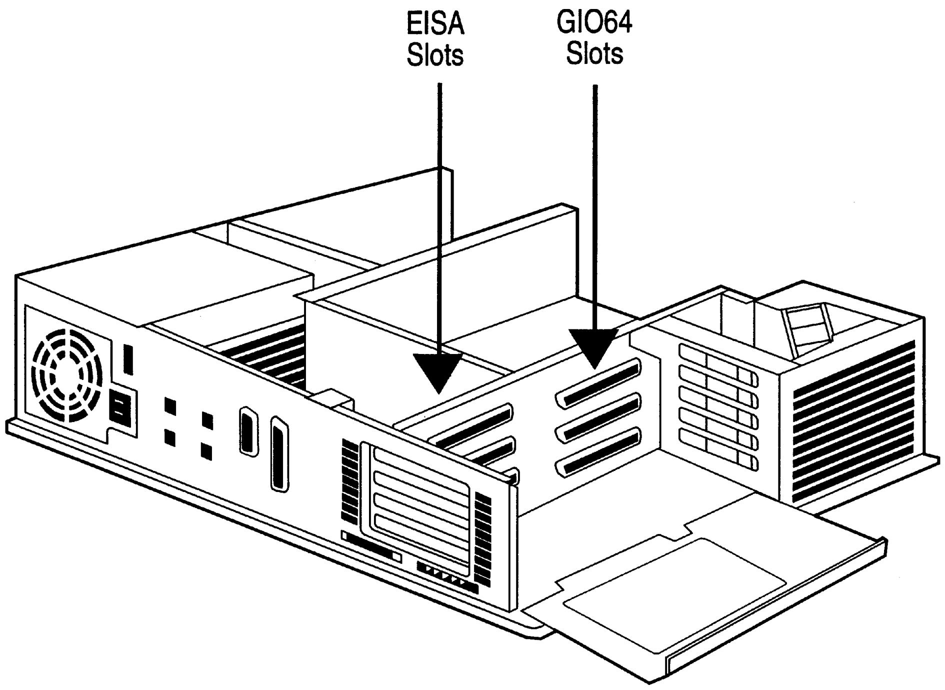


Figure 4-16 Open the I/O Cage Door

6. Note the location of the backplane, which appears directly in front of you when the I/O cage door is open.

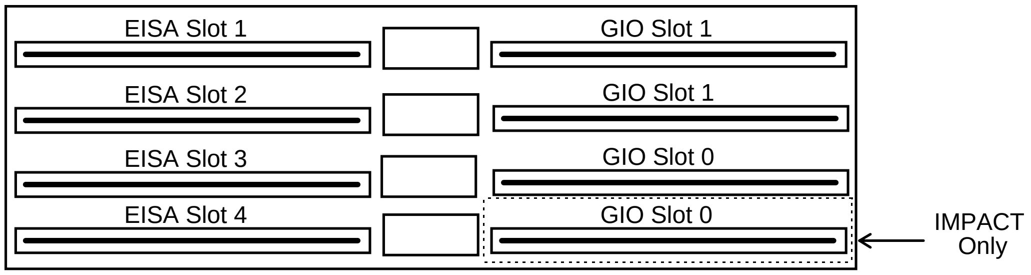


Figure 4-17 The Indigo2 EISA/GIO Backplane

SGI Indigo2 systems have three slots (four slots on the Indigo2 IMPACT). The top two GIO slots (and the bottom two GIO slots on the Indigo2 IMPACT) are wired together electrically. In this case, at most you can plug in two cards at one time.

Since NetFX adapters do not support slot-sharing, you can only connect one other card along with your NetFX adapter. Typically, SGI systems have a graphics card installed in slot 0.

If your system has a graphics card in slot 0, it's easiest to install your NetFX adapter in slot 1. However, if an Extreme™ graphics is installed in slot 0, move it to slot 1 and install your NetFX adapter in the bottom slot.

7. Write down the slot number you're going to use on the planning worksheet, then continue to the next procedure *Install the NetFX-GIO64 adapter*.

   -or-

   To remove a card to open a slot, complete the procedures in the section *Removing Installed Graphics Boards* (see page 4-3). After moving the board and writing down the slot number on the planning worksheet, continue to the next procedure *Install the NetFX-GIO64 adapter* below.

Install the NetFX-GIO64 adapter

If you are installing or replacing the GLM installed on an adapter that Prisa did not supply, follow the procedures in the section *Removing Installed Graphics Boards* in this chapter (see page 4-3).

1. Put on a properly-grounded, anti-static wrist strap, then remove the adapter from its anti-static bag.
2. Slide the NetFX adapter into the new slot position, aligning the bracket hole to the left of the slot. Gently push the card connector into the GIO64 slot until seated firmly (see Figure 4-18).
3. Insert and tighten the screw fastening the slot to the graphics board bracket.

4. If using an optical GLM, remove the protective cover from GLM cables. Attach cables to GLM at the back of computer. Otherwise keep optical GLMs covered to avoid dust.

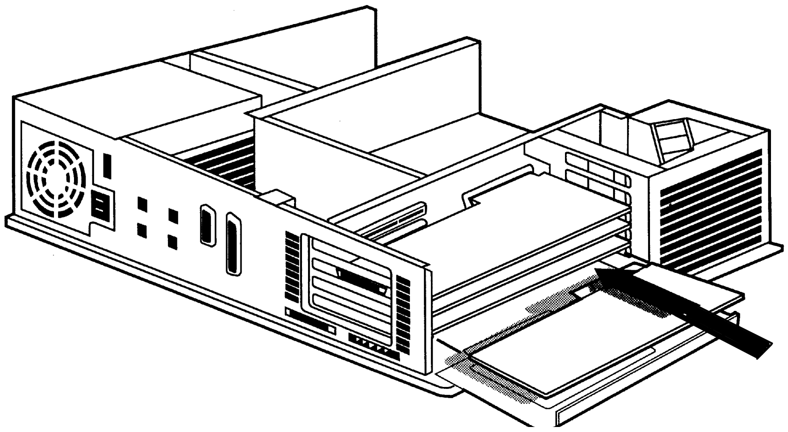


Figure 4-18 Install the NetFX Adapter

Replace the cover

1. Close the I/O cage door.
2. Place the rear edge of the cover behind the metal case, with the front edge of the cover tilting up. Look through the front of the case to align the four plastic tabs on the cover with the slots in the metal case. Pull the cover forward and insert the tabs into the slots (see Figure 4-19).
3. Lower the front of the cover and snap it into place.

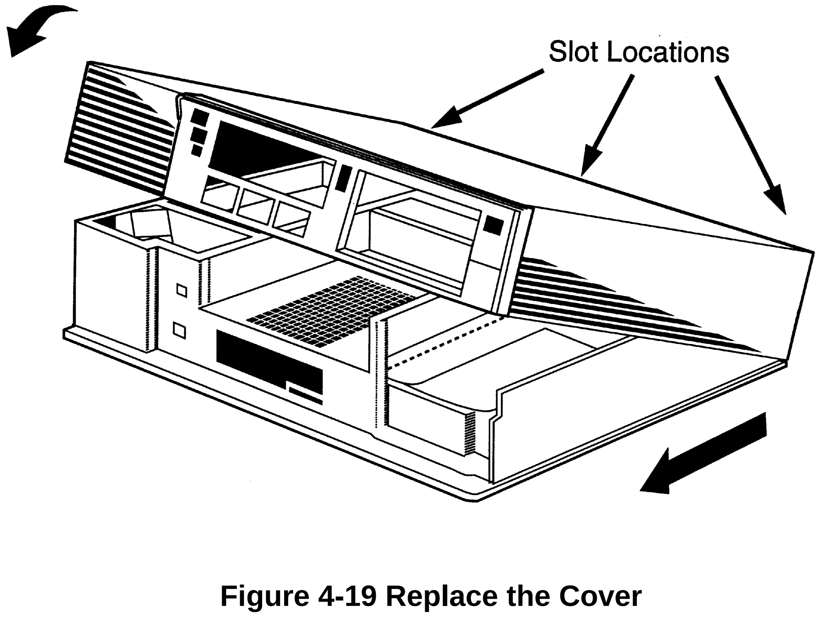


Figure 4-19 Replace the Cover

4. Snap the front faceplate onto the cover.
5. Install the metal security bar (if applicable, see Figure 4-13 on page 4-5):
   a. Slide the security bar into the front of the computer and out of a slot in the rear cover.
   b. Install the padlock through a hole in the security bar, then lock the security bar.
6. Close the faceplate door, then snap the faceplate onto the front cover.

   Adapter installation is now complete.

Turn on the system

1. Connect the power and channel cables to the back of the machine.
2. Press and release the power button on the machine’s front panel, then watch for a series of NetFX-based messages containing information about your new adapter.

The NetFX messages only appear for a few seconds. If you’re not sure the NetFX messages displayed, check the log file /var/adm/syslog. This log file records all of the information that appears on a monitor.

If the messages containing information about the new adapter do not appear, verify or repeat the NetFX-GI064 installation procedures. If the NetFX messages fail to appear a second time, run the installation diagnostics tests (see the section *Running Hardware Diagnostic Tests* in *Chapter 5*) and record the results before contacting Technical Support.

Where to go from here

➢ *Continue to* Chapter 5 Running Hardware Diagnostic Tests.

Installing the NetFX-HIO64 Adapter-

Before you begin, review the section *Guidelines.*You can choose from two HIO locations to install one or two NetFX-HIO64 adapters. If you need to remove a GLM, see the section *Remove a GLM from the NetFX-GIO64 or NetFX-HIO64* in this chapter (see page 4-18). If you are installing or replacing the GLM installed on the adapter, see *Removing Installed Graphics Boards* (see page 4-3). The NetFX-HIO64 adapter is designed to operate with the HIO bus found on the Silicon Graphics Onyx and Challenge DM, L, and XL systems. These are either deskside (see Figure 4-20) or rackmount (see Figure 4-21) systems.

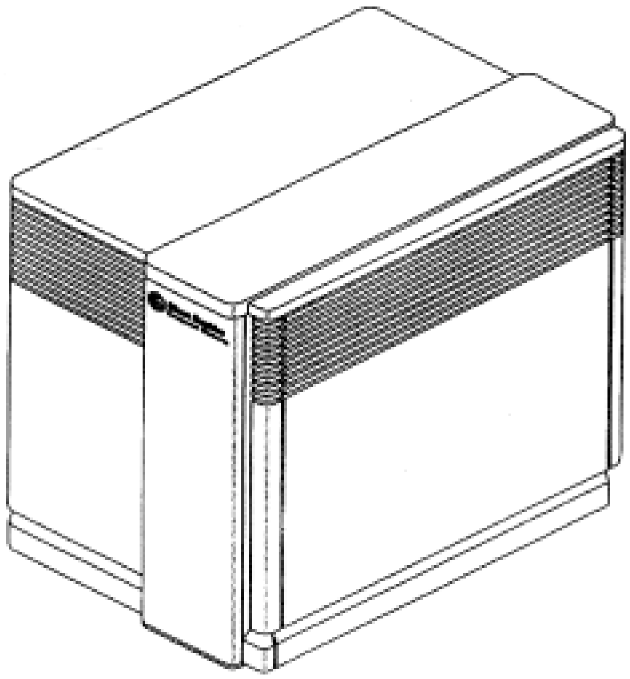


Figure 4-20 The Challenge L Deskside System

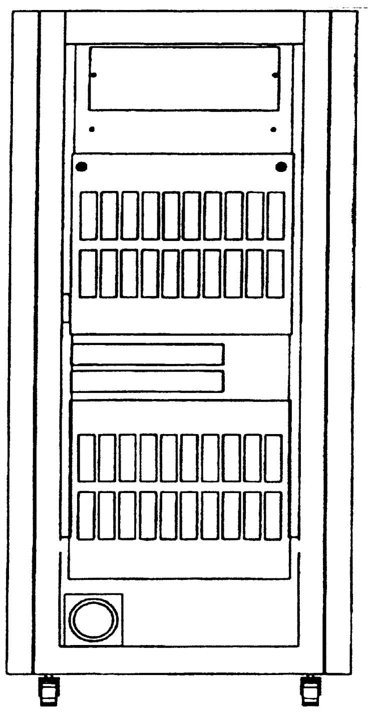


Figure 4-21 Open the Challenge XL Rackmount System

Turn off the system

Do not disconnect the power cable, as the cable forms a ground.

1. Turn off the key switch on the front panel.

   The system turns off automatically.

2. Turn off the AC circuit breaker.

Access the I/O panel

As you complete this procedure, be sure not to place tension on the internal cables while opening the I/O panel.

1. Open the cabinet cover (see Figure 4-22).

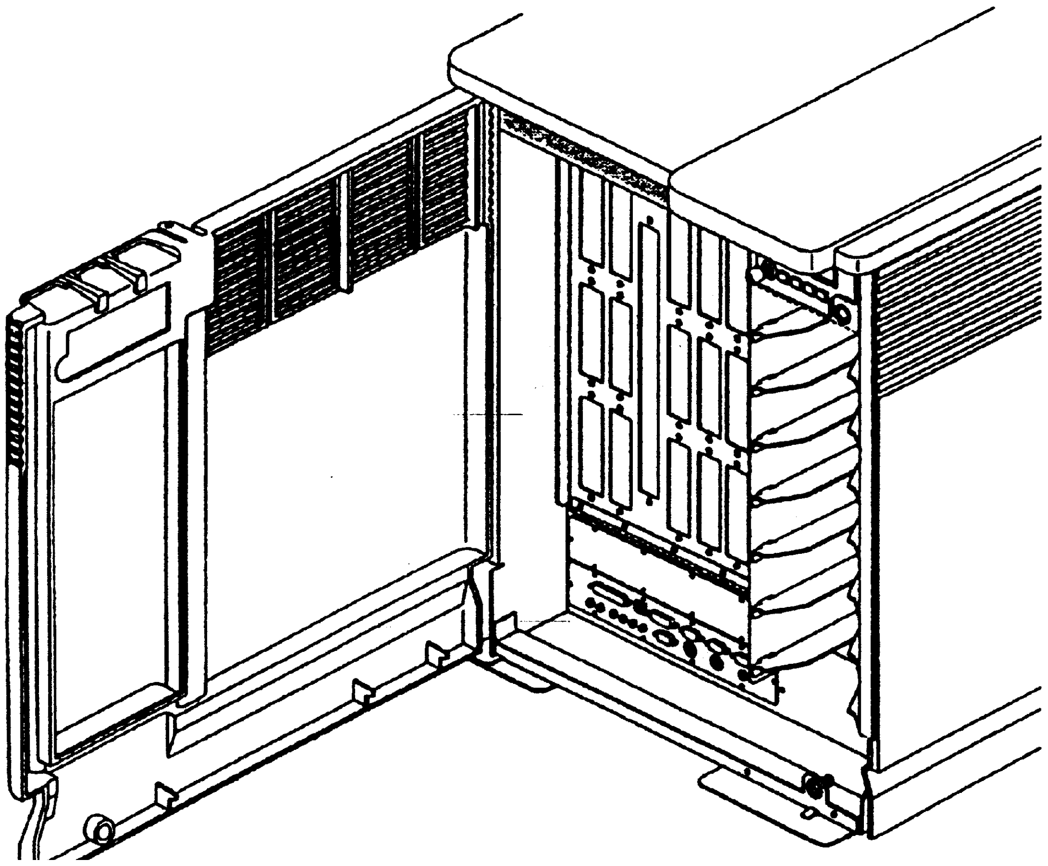


Figure 4-22 Open the Cabinet Cover

2. Remove the I/O panel screws (see Figure 4-20 for the deskmount system) or release the two quarter-turn fasteners (see Figure 4-21 for the rackmount system).

   Screws and fasteners are located along the top, and the panel is hinged at the bottom.

3. Grasp the top of the I/O panel and pull back carefully to rotate the panel open and reveal the card cage.

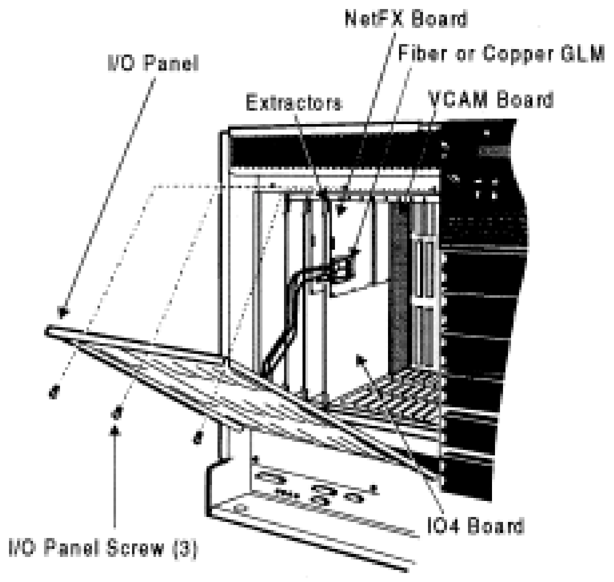


Figure 4-23 Open the I/O Panel (Deskside Cabinet)

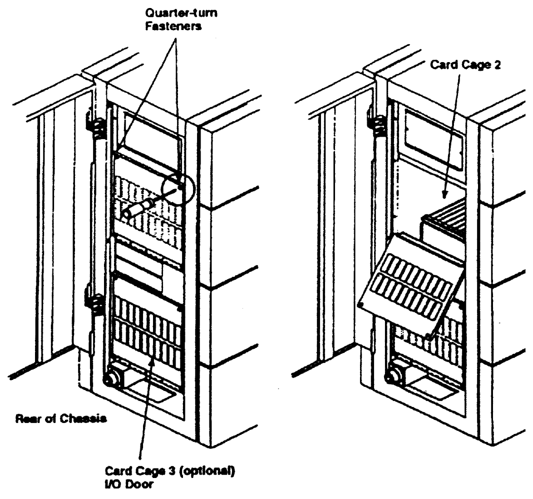


Figure 4-24 Open the Rackmount I/O Panel

Remove the air shield

1. Loosen the two screws on each one of the two retainer bars (see Figure 4-23).
2. Move the sliders holding the retainer bars, then remove the retainer bars.
3. Remove the air shield.

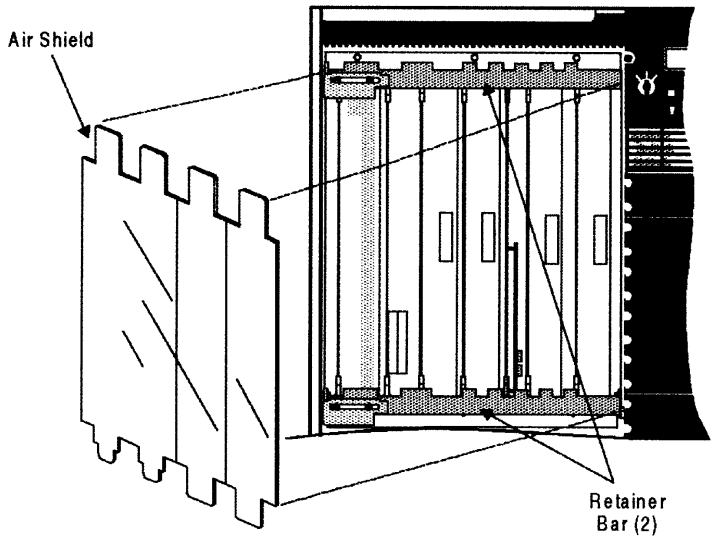


Figure 4-25 Remove the Air Shield

Remove an IO4 board

To make it easier to reassemble the unit, note the location of the cables before you start to remove any boards. Complete this procedure for each IO4 board you want to remove.

1. Put on a properly-grounded, anti-static wrist strap.
2. Prepare an anti-static surface large enough for the boards you are removing.

   If you still have them, use the anti-static enclosure bags in which the boards were shipped.
3. Disconnect the internal cables from the IO4 board.
4. Grasp both extractors and spread open to unseat the IO4 board. Next, remove the board from the card cage.
5. Place the board on the anti-static surface you prepared.

Install the NetFX-HIO64 adapter

1. Put on a properly-grounded, anti-static wrist strap.
2. Using the finger holds, hold the adapter above the IO4 board so that the four mounting holes align with the IO4 spacer columns (standoffs).
3. Firmly, but carefully, press the NetFX-HIO64 adapter onto the 100-pin connector.
4. Install the binding head screw fasteners into the standoffs.

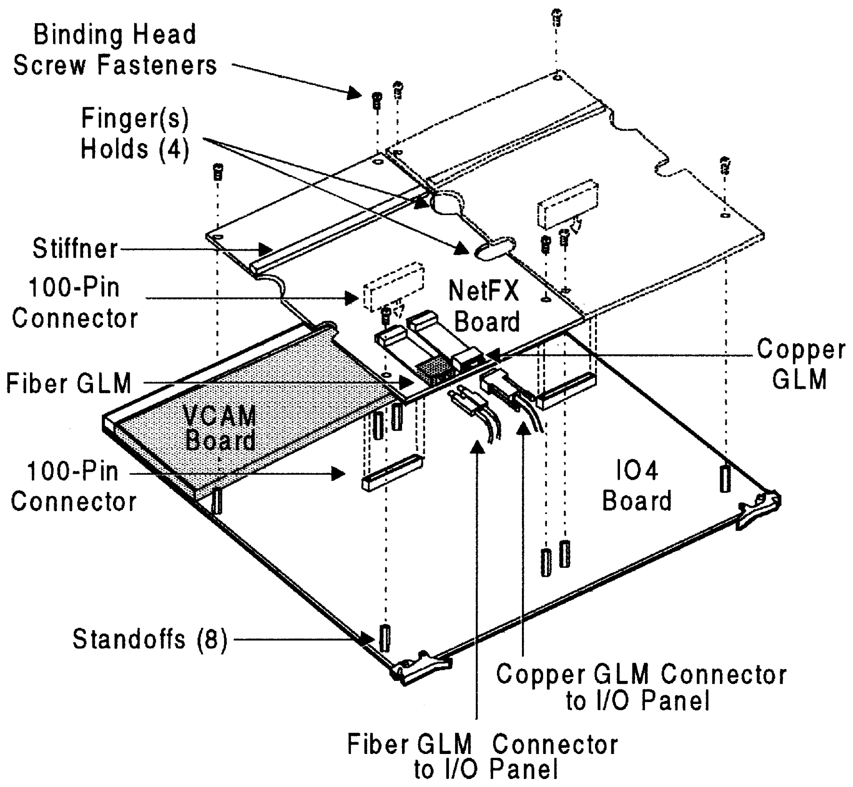


Figure 4-26 Install the NetFX-HIO64 Adapter Onto the IO4 Board

Install the IO4 board -

1. Reinstall each IO4 board into the correct slot, making sure the board is right side up (see Figure 4-26).
2. Reconnect the internal cables on the IO4 board.

Install the extender cable assembly

This procedure explains how to add an extender cable assembly between the external cable and the GLM (rather than connecting the external cable directly to the GLM). Prisa supplies an extender cable assembly for each GLM shipped.

1. For each extender cable assembly, unscrew the fasteners to remove the blank plates from the I/O panel (see Figure 4-27).
2. Remove each extender cable assembly from the packaging.

An extender cable assembly consists of an I/O panel plate, a connector (DB9 for copper connector or SC duplex for fiber optic connector), and a short connecting cable.

3. Screw the assembly I/O panel plate to the I/O panel.

4. For optical GLMs, remove the small protective caps on the internal assembly connector and the protective cover on the GLM that you're connecting.
5. Plug the extender cable assembly connector into the GLM connector.

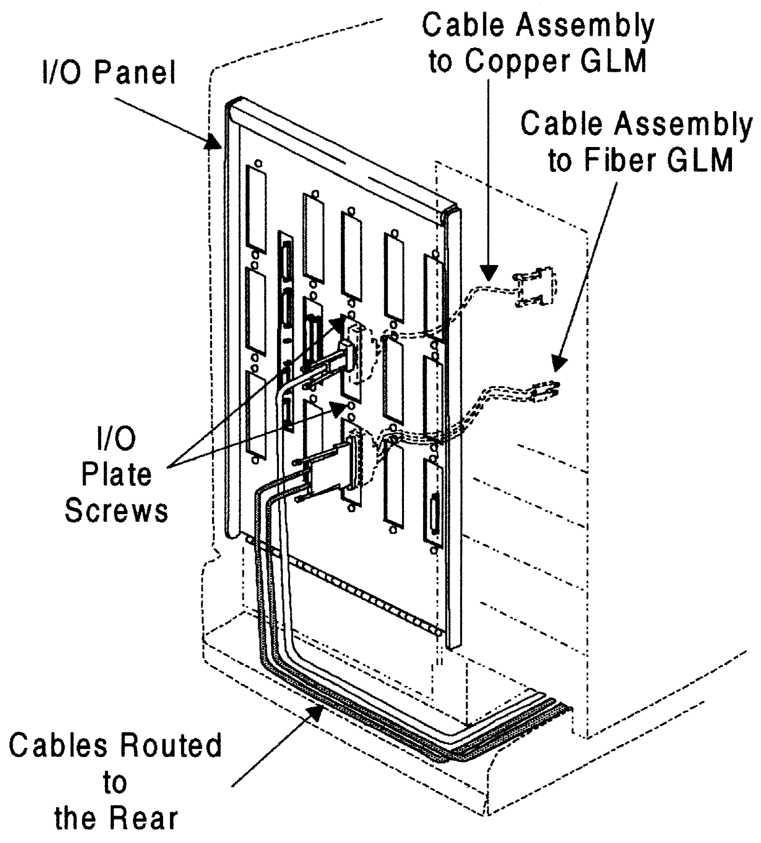

Cable Assembly to Copper GLM
I/O Panel
Cable Assembly to Fiber GLM
I/O Plate Screws
Cables Routed to the Rear

Figure 4-27 Install the Cable Assembly (Deskside System)

Reassemble the unit (deskside and rackmount configuration)

To avoid damaging the extender cable assembly, do not pinch cables while closing the I/O panel.

1. Reinstall the air shield (see Figure 4-25).
2. Gently close the I/O panel, making sure not to pinch the cables.
3. Fasten the I/O panel (see Figure 4-23 and Figure 4-24).
4. Plug the external data cable or cables into the I/O panel and route the cables (see Figure 4-27).
5. For optical connectors, remove the small protective caps on the external assembly connector and the protective cover on the external cable connector.

6. Plug the external cable into the external assembly connector on the I/O plate.
7. Close the cabinet cover.

Adapter installation is now complete.

Turn on the system

1. Turn on the AC circuit breaker.
2. Turn on the key switch on the front panel.

   When you add or change hardware, the boot sequence stops and a menu that lists several options appears.

3. From the menu, choose Enter command monitor.
4. Update information about your machine.

```
# update
```

5. Quit.

```
# exit
```

The menu displays again.

6. Choose Start system.

   The NetFX messages only appear for a few seconds. If you didn’t see them, check the log file /var/adm/syslog.

7. If the messages containing information about the new adapter do not appear, verify or repeat the NetFX-HI064 installation procedures. If the NetFX messages fail to appear a second time, run the installation diagnostics tests (see *Chapter 5 Running Hardware Diagnostic Tests*) and record the results before contacting Technical Support.

Where to go from here

➢ *Continue to* Chapter 5 Running Hardware Diagnostic Tests.

Installing the NetFX-PCI32 Adapter-

Before you begin, review the section *Guidelines* at the beginning of this chapter.

Unlike the GIO and HIO systems, Silicon Graphics includes complete documentation with all of the PCI systems that it delivers. For detailed information about your system, see the SGI-supplied *Workstation Owner's Guide.*

Install the NetFX-PCI32 adapter

1. Follow the instructions for installing a NetFX-PCI32 adapter in the SGI *Workstation Owner's Guide* for your computer.

2. Turn on your system according to the instructions in the SGI *Workstation Owner's Guide* for your computer. As the machine boots up, watch for a series of messages containing information about your new adapter.

   The NetFX messages only appear for a few seconds. If you're not sure the messages displayed, check the log file /var/adm/SYSLOG.

   If messages containing information about the adapter do not appear, verify or repeat the NetFX-PCI32 installation procedures. If the NetFX messages do not appear a second time, run the diagnostic tests (see *Chapter 5 Running Hardware Diagnostic Tests*) and record the results before contacting Technical Support.

Where to go from here

➢ *Continue to* Chapter 5 Running Hardware Diagnostic Tests.

Removing and Installing GLMs (Gigabaud Link Modules)

Do not remove GLMs from NetFX-PCI32 adapters. This section applies to NetFX-GIO64 and NetFX-HIO64 adapters only.

If you purchased your GLM from Prisa, your NetFX adapter is shipped with the GLMs installed. However, if you purchase your GLM from another supplier, or if you change from using copper media to fiber optics, you need to replace or install the GLM.

Remove a GLM from the NetFX-GIO64 or NetFX-HIO64

1. Remove the Fibre Channel cable.

2. Remove the NetFX adapter from the computer by reversing the procedures outlined in the sections *Install the NetFX-GIO64 adapter* (see page 4-8) or *Install the NetFX-HIO64 adapter* (see page 4-14).

3. Remove two screws from the bottom of the board that attach the GLM to the board.
4. On the underside of the adapter, gently spread the GLM feet to unsnap it from adapter.
5. Pull hinged lever on GLM up to separate GLM from adapter.
6. Gently disconnect and remove the GLM.
7. Replace the protective cover on the GLM connector.

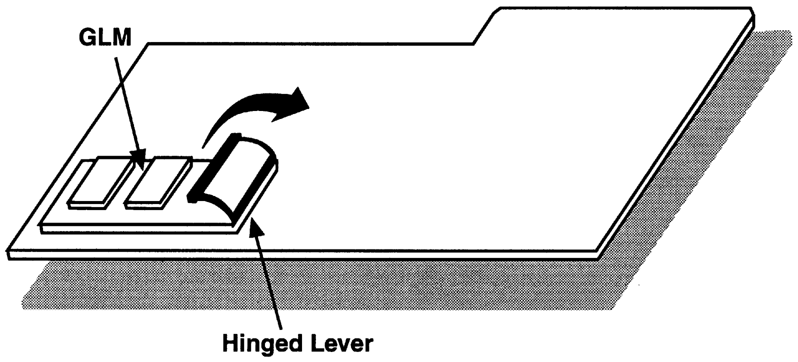


Figure 4-28 Remove the GLM

Install a new GLM

As you attach the GLM, hold it and the board at an angle; this makes it easier to line up the connection points properly.

1. If installing a new Optical GLM, remove the protective cover from the end of GLM and install the fiber optic cable.
2. Ensure GLM hinged lever is laying flat.

   Do not use the lever to snap the GLM in place.
3. Align GLM and board, connector-to-connector, and gently push until the GLM is attached properly.
4. Gently press the center of the GLM to snap its feet into place through the adapter.
5. Install two screws on the bottom of the board to attach the GLM to the board.

Where to go from here

➢ Continue to one of these procedures:

Install the NetFX-GIO64 adapter (see page 4-8).

Install the NetFX-HIO64 adapter (see page 4-14).

Install the NetFX-PCI32 adapter (see page 4-18).

5 Running Hardware Diagnostic Tests

NetFX at a Glance

1. Design a Fibre Channel Network.
2. Install NetFX software.
3. Install the adapter.
4. **Run diagnostic tests.**
5. Configure NetFX software, see Chapter 6.

Overview

This chapter explains how to run internal Pass/Fail diagnostic tests to verify the NetFX adapters are installed correctly. Before you begin, complete the procedures in *Chapter 3 Installing NetFX Software* and *Chapter 4 Installing NetFX Hardware.*

This chapter includes these sections:`

Guidelines

Keep these guidelines in mind as you prepare to run hardware diagnostics:

- Run diagnostics *before* you configure the software.
- Complete the test procedures in the order specified.

- If testing a dual channel adapter, you need to know which channel is Channel 1 and which is Channel 2 when plugging in the loopback shroud for channel-level tests.
- It takes around 20 minutes to run diagnostic tests for each channel.

Procedures

This section contains step-by-step procedures that explain how to run diagnostic tests.

Turn off the chkconfig flags

This procedure requires you to reboot the machine in which you installed the adapter. If you're using a shell window to view the progress of the diagnostics, reboot the machine containing the new adapter and *not* your workstation.

1. Login as **`root`**.
2. Turn off chkconfig netfx.

```
# chkconfig netfx off
```

3. Turn off chkconfig fsd.

```
# chkconfig fsd off
```

4. Reboot the system.

```
# /etc/reboot
```

A boot message indicates the NetFX adapter's WWN, slot number, and position.

```
NetFX slot 5 adapter 5 [2-D-00170]
NetFX slot 5 adapter 5 channel 1 present rev 2 wwn 1000006077002382
NetFX slot 5 adapter 5 channel 2 present rev 2 wwn 1000006077002383
```

Run internal diagnostic tests

This menu-driven utility guides you through a series of tests that run internal Pass/Fail diagnostics on NetFX adapters. It also tests the ability of a channel to send and receive, which requires a loopback shroud.

1. Plug the loopback shroud into the channel you're testing.

To determine the channel number, check the adapter's layout. Adapter layouts are in the section *Determining Adapter Attributes* of Chapter 2.

2. Login as **`root.`**

3. Start the diagnostic utility.

```
# /usr/NetFX/bin/netfxdiag
```

A message containing information about installed adapters, including the slot number for installed boards, appears.

```
NetFX Diagnostic Software Copyright
(c) 1995, 1996, 1997 Prisa Networks, Inc.

NetFX Single Channel GIO64 board detected in GIO slot 1

/tmp/netfxinstall.log opened

NetFX Installation Test
1) Test Single Channel board in slot 1
Q) quit NetFX diagnostics

Enter your selection:
```

Installed board's slot number

Figure 5-1 The First NetFX Diagnostics Screen

4. Enter the value that corresponds to the test you want to run.

A summary of test results appears.

```
running: Test Single Channel board in slot 1     To stop test press <!>

Testing board in slot: 1

running Test Dual Channel board in slot 1        To stop test press <!>
running: Board ID detect                                   PASSed
running: CSR Reset                                         PASSed
running: CSR Set Clear                                     PASSed
running: CSR Set Reset                                     PASSed
running: Read Nvram                                        PASSed
running: Nvram Write Disable                               PASSed
running: write delay access                                PASSed

Slot 0: Ch. 1
WWN: 00.00.14.24.00.67.00.00
```

Shortened World Wide Number

Figure 5-2 The Second NetFX Diagnostics Screen

Next, a message identifies the channel being tested.

```
Slot 1: Ch. 1
Place loopback shroud on channel 1 of board in slot 1.
Press a key to begin.
```

Figure 5-3 The Top of the Third NetFX Diagnostics Screen

5. Press any key to start the tests.

A summary of the test results appears.

```
running: FIFO test                          PASSed
running: EPLD test                          PASSed
running: Read local memory                  PASSed
running: Tachyon present                    PASSed
running: Tachyon reset                      PASSed
running: LM:Address pattern                 PASSed
running: LM:Inverse Address                 PASSed
running: Ext.Small/fast local to local      PASSed
: All detected boards                       PASSed

Press any key to exit.
```

Figure 5-4 The Bottom of the Third Diagnostics Screen

6. Exit the diagnostics program, press **Q**.
7. Remove the loopback shroud and reinsert the channel cable into the adapter board.

Hardware diagnostics are complete.

Turn on the chkconfig flags

1. Turn on chkconfig.

```
# chkconfig netfx on
```

2. Turn on chkconfig.

```
# chkconfig fsd on
```

3. Reboot the system.

```
# /etc/reboot
```

Check the LED indicators

LED (Light Emitting Diode) indicators display a green light to indicate a good channel connection, which means the adapter can receive a recognizable Fibre Channel data stream. If the LED indicators are not lit, it means a good connection has not been made.

- ➢ Check the LED indicators on the LED face plate, described in the section *Determining Adapter Attributes* of Chapter 4, to ensure the adapter is installed correctly and a valid connection has been made.

Resolve diagnostic test failures

Complete these steps if any of the diagnostic tests fail.

1. Verify or repeat the hardware installation procedures.
2. Rerun the diagnostic utility.
3. Write down the test results.
4. Contact the technical support representative at the company that supplied your Prisa product.

Where to go from here

- ➢ Continue to Chapter 6 Configuring NetFX Software.

6 Configuring NetFX Software

NetFX at a Glance

1. Design a Fibre Channel Network.
2. Install NetFX software.
3. Install the adapter.
4. Run diagnostic tests.
5. **Configure NetFX software.**

Overview

This chapter includes these sections:

Guidelines

As you configure NetFX Software, keep these guidelines in mind:

- The NetFX Configuration files are case-sensitive, which means you need to type information exactly as shown. For example, if instructed to type NetFX, type NetFX and not netfx.
- When using the backslash "\" character for line continuation, make sure that this is the last character on the line. Any kind of whitespace character such as a space or a tab can cause problems with line continuations.

It's also a good idea to have these Silicon Graphics documents available:

- *IRIX Admin: Disks and File Systems*
- *IRIX Admin: Networking and Mail*

Relevant Terminology

NLPort - Node Loop Port, a node on a Fibre Channel Arbitrated Loop (FC-AL)

NPort - Node Port, a node on a Fibre Channel Switch Fabric

Each NLPort or NPort can be a system, a RAID controller, or a port on an individual Fibre Channel disk drive, and has the following attributes:

WWN - World Wide Name. This is the hardware address for an N/NL Port. It is a 64 bit (8 byte) value. The first 5 bytes are fixed for NetFX adapters, at 10.00.00.60.77. The last 3 bytes must be unique on a given system, and are assigned to each board by the manufacturer. For example, two NetFX adapter ports could have the World Wide Names 10.00.00.60.77.00.00.01 and 10.00.00.60.77.00.00.02. A WWN must be unique for each port on a local system as well as each port in a topology; two systems plugged into the same loop or fabric cannot have the same WWN.

Channel ID - An arbitrary non-zero hexadecimal number assigned to each N/NL Port. Each local Fibre Channel port on a system must have a unique Channel ID. When we enter information for a local port in /etc/NLPorts, we must assign it a Channel ID unique to that system. Subsequently when we enter information for a remote port in /etc/NLPorts, we must specify the Channel ID of the local port that we must use to reach that remote port.

Loop ID - An arbitrary hexadecimal number between 0 and 7d assigned to each N/NLPort. Each Loop ID maps to a unique Arbitrated Loop Physical Address (AL_PA). There is a Loop ID->AL_PA mapping for each Loop ID in the Loop ID Reference Table in Appendix B. When we enter information for a local port in /etc/NLPorts, we must choose a Loop ID that no other device on the Loop is using. When we enter information for a remote port in /etc/NLPorts, we need to make sure that we use the Loop ID that this device has been configured to use.

NPname - An arbitrary text name assigned to a N/NL Port. It can be up to 256 characters, and must be unique in each /etc/NLPorts file; two systems plugged into the same loop or fabric cannot have the same NPname.

NPalias - An additional arbitrary text name assigned to a N/NL Port. A NPalias can be up to 256 characters, and functions just like an IP alias; it is an alternate name for a port, but **it is not required**.

Configuration Files

/etc/NLPorts

This is where all of our Fibre Channel N/NL Ports are defined, including the local ports of the host system whose file you are editing, and the remote ports that this system will communicate with via its local ports.

Each line in /etc/NLPorts denotes one Fibre Channel port, and is entered in the following format:

```
<WWN>   <Channel ID>,<Loop ID>   <NPname>   <NPalias>
```

See the *Relevant Terminology* section for explanations of the fields.

NOTE the following guidelines:

- In **SWITCH** configurations, **ALWAYS** use **ZERO** as a **Loop ID**
- **NEVER** use **ZERO** as a **Channel ID** in **ANY** configurations

/etc/config/netfx.options

This file is where we define:

- local adapters (type, slot, position)
- local channels/ports for each local adapter (enable/disable, npname, protocol selection, nport options, ip options)
- protocols are SCSI (si), IP (ip), and Transporter (tpt)

NOTE the following guidelines:

- All of the information that you must enter for this file can be found in the `/var/adm/SYSLOG` file.
- For each port described here, you **MUST** use the npname of the port in /etc/NLPorts that you wish to configure.
- NPort options are explained in the man page for `tnportcfg`.
- IP options are explained in the man page for `fcipcfg`.
- Always use override=yes to use the WWN specified in /etc/NLPorts instead of reading the NVRAM on the adapter.

/etc/config/fsd.options

Here we define the disk drives that we will communicate with directly, with the following fields:

- `unit number` - this is an arbitrary number, and must be unique for each disk device.

- `npname` - for each disk device describes here, you **MUST** use the npname of the disk device port (RAID or individual drive) in /etc/NLPorts that you wish to configure.
- `device name` - this is the name of the device file that will be created for each disk device.
- `logical unit number(LUN)`- this is in 64 bit SCSI-3 format
    - e.g. an 8 bit lun of 02 would be a 64 bit LUN of 0x0002000000000000
- `additional options` - Additional fsd options (as well as information about configuring LUNs) are explained in the man page for `fsdcfg`.

/etc/xlvgdisks

xlvgdisk is the Prisa xlv generic disk driver. It allows SCSI disk devices configured with NetFX SCSI Disk Driver software to be used in SGI's XLV logical volumes by mapping NetFX fsd devices to xlvgdisk devices. Each entry is comprised of

- `unit number` - a unique decimal non-zero value.
- `device name` - the device name from /etc/config/fsd.options that will be mapped to a xlvgdisk device.

For more information on using NetFX Fibre Channel disk devices with XLV, see the man page for `xlvgcfg` and the *Using NetFX Software* section.

/etc/hosts

This is the file used for configuring NetFX Fibre Channel ports as IP interfaces. Each local Fibre Channel port that we wish to use as an IP interface must have an entry in /etc/hosts. The IP interface name must match the port's npname in /etc/NLPorts. Each remote IP port can be defined either in /etc/hosts or by using a name service protocol such as NIS or DNS. However, keep in mind that the name in the NIS or DNS maps must still match the entry for that remote port in /etc/NLPorts.

Steps to Configure NetFX Fibre Channel

1. Create Topology Diagram
2. Edit /etc/NLPorts
3. Edit /etc/config/netfx.options
4. Edit /etc/config/fsd.options (as necessary)
5. Edit /etc/xlvgdisks (as necessary)
6. Edit /etc/hosts (as necessary)
7. Complete Configuration

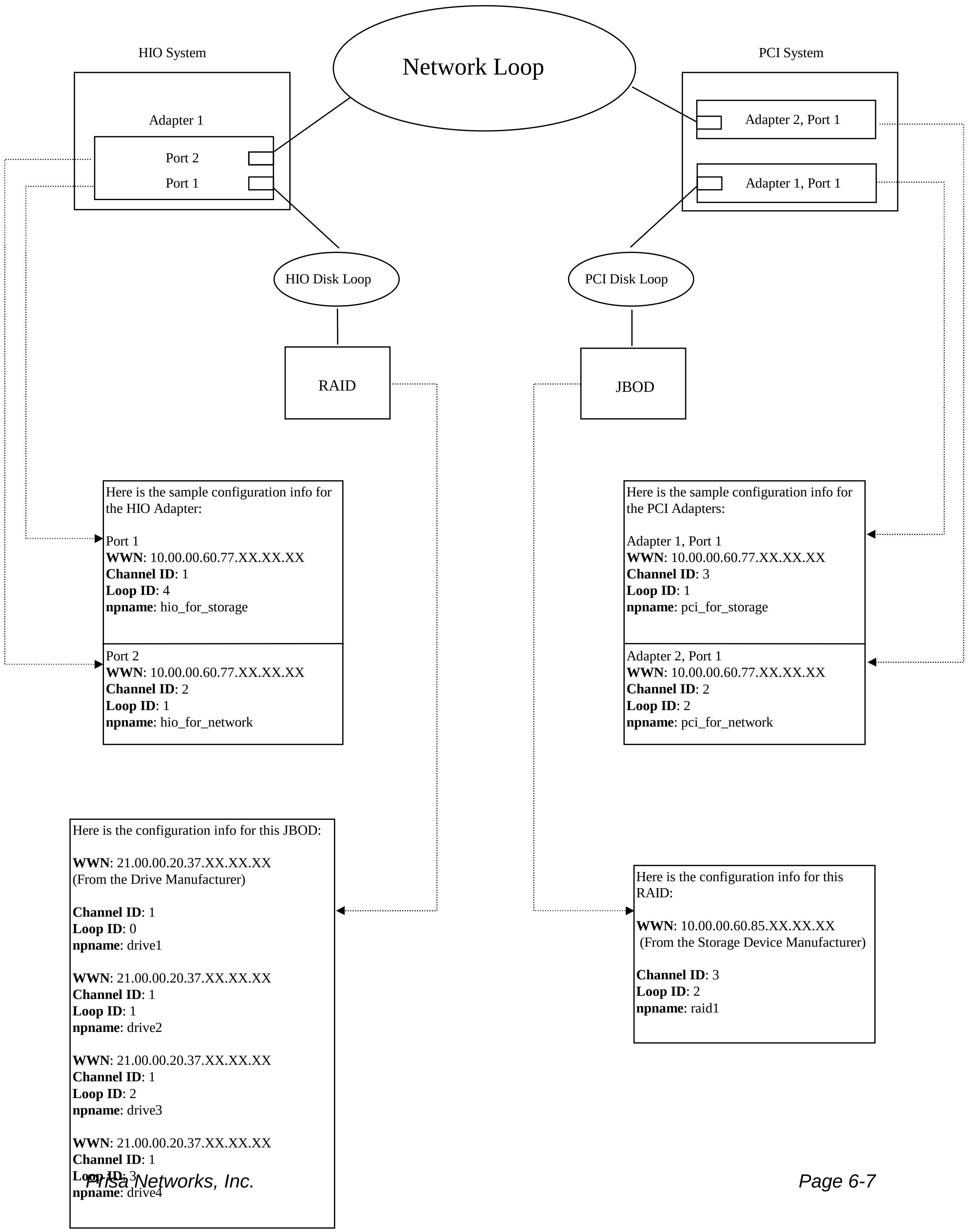
Example Configuration

Here is an example configuration. We will look at both systems for each step.

1. Topology Diagram

When configuring NetFX software, it is always best to create a diagram of your Fibre Channel Topology before you edit the configuration files, to have an overall view of the configuration. For our example configuration, we have two systems: a system with a dual channel hio NetFX adapter, and a system with two NetFX pci adapters. Each system uses one port for networking and one for storage, both arbitrated loops.

NOTE: The "XX.XX.XX" in all WWN examples represent real numbers that will be assigned to your NetFX adapter or disk device. You MUST use these actual numbers. DO NOT put "XX.XX.XX" in any of your config files!!!!



Here is the sample configuration info for the HIO Adapter:

Port 1
WWN: 10.00.00.60.77.XX.XX.XX
Channel ID: 1
Loop ID: 4
npname: hio_for_storage

Port 2
WWN: 10.00.00.60.77.XX.XX.XX
Channel ID: 2
Loop ID: 1
npname: hio_for_network

Here is the sample configuration info for the PCI Adapters:

Adapter 1, Port 1
WWN: 10.00.00.60.77.XX.XX.XX
Channel ID: 3
Loop ID: 1
npname: pci_for_storage

Adapter 2, Port 1
WWN: 10.00.00.60.77.XX.XX.XX
Channel ID: 2
Loop ID: 2
npname: pci_for_network

Here is the configuration info for this JBOD:

WWN: 21.00.00.20.37.XX.XX.XX
(From the Drive Manufacturer)

Channel ID: 1
Loop ID: 0
npname: drive1

WWN: 21.00.00.20.37.XX.XX.XX
Channel ID: 1
Loop ID: 1
npname: drive2

WWN: 21.00.00.20.37.XX.XX.XX
Channel ID: 1
Loop ID: 2
npname: drive3

WWN: 21.00.00.20.37.XX.XX.XX
Channel ID: 1
Loop ID: 3
npname: drive4

Here is the configuration info for this RAID:

WWN: 10.00.00.60.85.XX.XX.XX
(From the Storage Device Manufacturer)

Channel ID: 3
Loop ID: 2
npname: raid1

2. Edit /etc/NLPorts

HIO System

Before we edit our /etc/NLPorts file, we need to get the WWNs and Loop IDs for our N/NLPorts. For our local ports, we look in the /var/adm/SYSLOG file, which has output similar to this:

```
unix: NetFX slot 5 adapter 5 [2-D-00170]
unix: NetFX slot 5 adapter 5 channel 1 present rev 2 wwn 1000006077XXXXXX
unix: NetFX slot 5 adapter 5 channel 2 present rev 2 wwn 1000006077XXXXXX
```

NOTE: The [2-D-00170] is [Number of Channels-Hardware Rev-Serial Number]

For remote ports, we have to check whatever method is available from that device's manufacturer to get the WWNs. NOTE: when looking at Seagate Barracuda Fibre Channel Drives, you will see a number like 00.20.37.XX.XX.XX. You must add a 21.00 or 22.00 to the front of this number, depending on which port of the disk you would like to use. This usually corresponds to the Port 1 and Port 2 or Port A and Port B on the JBOD that you connect to your NetFX Adapter.

The Loop IDs for the local ports are assigned by us. We must make sure that the Loop IDs we assign are not in conflict with any remote ports. We must get the Loop IDs for the remote ports from the remote devices. For remote systems running NetFX software, we check the /etc/NLPorts file on each remote system; for remote disk devices, we use whatever method is available from that device's manufacturer (e.g. Most JBOD manufacturers have a method of setting the Loop IDs on their chassis according to the slots).

The /etc/NLPorts file for the HIO system looks like this:

```
# My Local Adapter and 4 Drives in a JBOD, on Channel 1
10.00.00.60.77.XX.XX.XX       1,4       hio_for_storage
21.00.00.20.37.XX.XX.XX       1,0       drive1
21.00.00.20.37.XX.XX.XX       1,1       drive2
21.00.00.20.37.XX.XX.XX       1,2       drive3
21.00.00.20.37.XX.XX.XX       1,3       drive4

# My Local Adapter and a remote system, on Channel 2
10.00.00.60.77.XX.XX.XX       2,1       hio_for_network
10.00.00.60.77.XX.XX.XX       2,2       pci_for_network
```

NOTE: If I were accessing the remote system through a **Switch**, the /etc/NLPorts entries for Channel 2 would look like this:

```
# My Local Adapter and a remote system, on Channel 2
10.00.00.60.77.XX.XX.XX       2,0       hio_for_network
10.00.00.60.77.XX.XX.XX       2,0       pci_for_network
```

PCI System

Again, we need to get the WWNs and Loop IDs for our N/NLPorts. For our local ports, we again look in the /var/adm/SYSLOG file, which has output similar to this:

```
unix: NetFX adapter PCI slot 1
unix:     [Part Number:5526 | Revision:COO | Serial Number:830371]
unix: pnetfx: pci32 slot 1 position 5 channel 1 present, rev 2 WWN 10.0.0.60.77.XX.XX.XX

unix: NetFX adapter PCI slot 1
unix:     [Part Number:5526 | Revision:COO | Serial Number:830372]
unix: pnetfx: pci32 slot 1 position 7 channel 1 present, rev 2 WWN 10.0.0.60.77.XX.XX.XX
```

The Loop IDs for the local ports are assigned by us. We must make sure that the Loop IDs we assign are not in conflict with any remote ports. We must get the Loop IDs for the remote ports from the remote devices. For remote systems running NetFX software, we check the /etc/NLPorts file on each remote system; for remote disk devices, we use whatever method is available from that device's manufacturer (e.g. Most JBOD manufacturers have a method of setting the Loop IDs on their chassis according to the slots).

The /etc/NLPorts file for the PCI system looks like this:

```
# My Local Adapter and 1 RAID, on Channel 3
10.00.00.60.77.XX.XX.XX       3,1       pci_for_storage
21.00.00.60.85.XX.XX.XX       3,2       raid1

# My Local Adapter and a remote system, on Channel 2
10.00.00.60.77.XX.XX.XX       2,1       hio_for_network
10.00.00.60.77.XX.XX.XX       2,2       pci_for_network
```

NOTE: Having the disk channel be Channel 3 is just fine. The Channel ID can be any NON-ZERO number, as long as everything on that Channel has the same Channel ID.

3. Edit /etc/config/netfx.options

HIO System

Before we edit our /etc/config/netfx.options file, we need to get the adapter information from /var/adm/SYSLOG file, which has output similar to this:

```
unix: NetFX slot 5 adapter 5 [2-D-00170]
unix: NetFX slot 5 adapter 5 channel 1 present rev 2 wwn 1000006077XXXXXX
unix: NetFX slot 5 adapter 5 channel 2 present rev 2 wwn 1000006077XXXXXX
```

From this information, we enter the following information in /etc/config/netfx.options:

```
adapt type=hio_dual slot=5 position=5
    chan port=1 enable=yes npname=hio_for_storage override=yes si=yes ip=no tpt=no
    chan port=2 enable=yes npname=hio_for_network override=yes si=no ip=yes tpt=no
```

NOTE: If we were using Channel 1 on a Switch, and we were using NetFX ZIP software on Channel 2 (still on a Loop), the entries would look like this:

```
 adapt type=hio_dual slot=5 position=5
    chan port=1 enable=yes npname=hio_for_storage override=yes si=yes ip=no tpt=no \
        nport_option=(-a topology=StaticLink) \
        nport_option=(-a NameServer=yes) \
        nport_option=(-a RT_TOV=100) \
        nport_option=(-a ED_TOV=5000)
    chan port=2 enable=yes npname=hio_for_network override=yes si=no ip=yes tpt=no \
        nport_option=(-a FastIP_Enable=yes)
```

PCI System

Again, we need to get our local adapter info from the /var/adm/SYSLOG file, which has output similar to this:

```
unix: NetFX adapter PCI slot 1
unix:    [Part Number:5526 | Revision:COO | Serial Number:830371]
unix: pnetfx: pci32 slot 1 position 5 channel 1 present, rev 2 WWN 10.0.0.60.77.XX.XX.XX

unix: NetFX adapter PCI slot 1
unix:    [Part Number:5526 | Revision:COO | Serial Number:830372]
unix: pnetfx: pci32 slot 1 position 7 channel 1 present, rev 2 WWN 10.0.0.60.77.XX.XX.XX
```

From this information, we enter the following information in /etc/config/netfx.options:

```
adapt type=pci32_single slot=1 position=5
    chan port=1 enable=yes npname=pci_for_storage override=yes si=yes ip=no tpt=no

adapt type=pci32_single slot=1 position=7
```

```
    chan port=1 enable=yes npname=pci_for_network override=yes si=yes ip=no tpt=no
```

4. Edit /etc/config/fsd.options

HIO System

Here we describe the four SCSI devices attached to the HIO system:

```
disk unit=1  npname=drive1  devname=dks777d1  lun=0
disk unit=2  npname=drive2  devname= dks777d2  lun=0
disk unit=3  npname=drive3  devname= dks777d3  lun=0
disk unit=4  npname=drive4  devname= dks777d4  lun=0
```

The NetFX software requires the devname field to be present, but ignores it and uses the device name dks777d<unit> instead. To eliminate confusion, have the devname match the name of the Fibre Channel disk device that will be constructed.

PCI System

Here we describe the four SCSI devices attached to the PCI system:

```
disk unit=1  npname=drive1  devname= dks777d1 lun=0
```

5. Edit /etc/xlvgdisks

HIO System

Here we describe the four SCSI devices attached to the HIO system that we want mapped to devices we can use with XLV. This file is required with IRIX versions earlier than 6.4.

```
1 dks777d1
2 dks777d2
3 dks777d3
4 dks777d4
```

6. Edit /etc/hosts

HIO and PCI Systems

Here we put the same two entries in /etc/hosts on both systems. This is because in our simple example we are not using any type of name service (e.g. DNS or NIS), so each system will need to have the other system's address as well as its own in its /etc/hosts file:

```
192.1.1.1      hio_for_network
192.1.1.2      pci_for_network
```

7. Complete configuration

1) Set the proper `chkconfig` flags:

   `# chkconfig netfx on`

   `# chkconfig` dks777d<unit> `on (`if using disk devices)

   `# chkconfig network on (`if using IP networking)

   `# chkconfig xlvgdisk on (`if using xlvgdisk)

2) Make sure disks or arrays are connected and on-line

3) Reboot the machine:

   ```
   # reboot
   ```

   As the machine boots up, a notice similar to this should appear (this is also recorded to /var/adm/SYSLOG:

```
NOTICE: hio_for_storage at NetFX slot 5 adapter 5 channel 2 up
```

8. Using NetFX Software

The following is information on using NetFX Software. It contains the following sections:

1) Using `fsdx` to partition NetFX disk devices
2) Using Fibre Channel Disk Devices with XLV
3) Using `fccp`

1) Using fsdx

The `fsdx` command manages the partitions that subdivide storage volumes on Fibre Channel SCSI disk devices. It is used to create or update volume header information, similar to the standard IRIX utility `fx`.

For example, to create a new volume header with (3) 4 gigabyte partitions for use with a XFS filesystem on a disk device defined as dks777d1 in `/etc/config/fsd.options`, we would use the following `fsdx` command:

```
# /usr/NetFX/bin/fsdx -d dks777d1 -N -m4g -txfs -p0 -p1 -p2
```

We can then use the standard IRIX command `prtvtoc` to view the volume header on the device:

```
# prtvtoc fsd1vol
```

NOTE: If you are on a system running IRIX 6.3 or below and do a "prtvtoc" on a drive that has been partitioned by a system running IRIX 6.4 and above, you will get a "Trace/BPT trap (core dumped)" message. This is an SGI issue, where the IRIX 6.4 and above systems write less label info, so when the IRIX 6.3 and below systems look for the label info, they get less info than they expect. You should still be able to mount a previously made filesystem or make a new filesystem on these volumes without a problem.

To create an XFS filesystem on a partition (partition 1 for example), we can use the standard IRIX command `mkfs` like so:

```
# mkfs /dev/dsk/fsd1s1
```

NOTE: The "s1" refers to "slice 1", another way to say "partition 1".

For more information on using disk devices with IRIX, see the SGI InSight™ book, **IRIX Admin: Disks and Filesystems**.

2) Using Fibre Channel Disk Devices with XLV

The xlv generic disk driver, xlvgdisk, allows non-SGI disk devices to be used in XLV logical volumes. If you want to use Fibre Channel disk devices in an XLV volume, you must provide an entry for each device in `/etc/xlvgdisks`. **IT IS NOT REQUIRED FOR SYSTEMS RUNNING IRIX 6.4 and HIGHER.** These systems can use the devices defined in /etc/config/fsd.options in the `xlv_make` file.

NOTE: See the man pages for `xlvgdisks`, `xlvgdisk`, `xlvgcfg`, and `xlv_make`, as well as the SGI InSight™ book, **IRIX Admin: Disks and Filesystems**, for more information on disk striping.

Format an entry in `/etc/xlvgdisks` as follows:

```
<unit number>  <disk device name>
```

Recall our entries in `/etc/xlvgdisks` for our HIO system:

```
1 dks777d1
2 dks777d2
3 dks777d3
4 dks777d4
```

With xlvgdisk chkconfig'd on, the next time we reboot the system `xlvgdisk` creates devices for use with XLV and maps them to the devices listed in `/etc/xlvgdisks`. For example, there will files in /dev/rdsk named `xlg0d1, xlg0d2, xlg0d3,` and `xlg0d4` that will map to dks777d1, dks777d2, dks777d3, and dks777d4 respectively. See the `xlvgdisk` man page for more info.

When the system comes up, log in as root and use `fsdx` (Prisa equivalent of IRIX fx) to make partitions on the first drive (In this example we use the `-r` option of `fsdx` to create one partition for the entire drive):

```
# /usr/NetFX/bin/fsdx -d dks777d1 -N -r -t xlv -p0
```

Now try to print a volume header for the first drive:

```
# prtvtoc dks777d1vol
```

You should get volume header info for the drive. Now try `fsdx` and `prtvtoc` for the other disk devices: dks777d2, dks777d3, and dks777d4. If either command hangs or gives "I/O Error" on any drive, recheck the info in the configuration files (see the Troubleshooting Guideline Storage section), reboot if you made any corrections, and try again.

Now, set up striping with SGI's XLV software:

Create a file called "xlv_makefile", and enter the following info:

```
vol jbod_volume1
data
ve -stripe -stripe_unit 256 xlg0d1s0 xlg0d2s0 xlg0d3s0 xlg0d4s0
end
exit
```

NOTE: a stripe unit of 256 has proven optimal for using up to 16 Seagate Barracuda 9GB Fibre Channel Disks. When using other disk devices (RAIDs, etc.), please see the SGI InSight™ book, **IRIX Admin: Disks and Filesystems** for more information on striping parameters.

Then, save the file and type the following:

```
# xlv_make xlv_makefile
```

Now, reboot your system.

When your system comes back up, login as root, and enter the following command:

`# ls /dev/dsk/xlv` (use `/dev/xlv` for IRIX 6.4 systems)

This should return a file of the form <system name>.<xlv volume name>

For example, the file on my system, which I named "alfa", is called alfa.jbod_volume1.

Now, make an XFS filesystem on this XLV volume.

```
# mkfs /dev/dsk/xlv/<system name>.jbod_volume1
```

NOTE: On systems running IRIX 6.3 and below, the XLV Volume device file `/<system name>.jbod_volume1` would be located in /dev/dsk/xlv and /dev/rdsk/xlv. On systems running IRIX 6.4 and above, however, this file would be located in /dev/xlv.

Once this completes, all that is left is to mount the filesystem. Create a mount point first:

```
# mkdir /jbod_xfs
```

Then mount your newly made filesystem:

```
# mount /dev/dsk/xlv/<system name>.jbod_volume1 /jbod_xfs
```

Now you are all set with a mounted XFS filesystem on your XLV logical volume.

3) Using FCCP

Fccp is a utility that uses the Prisa NetFX Transporter protocol to transfer files between systems. It has a simple syntax similar to the standard IRIX `rcp` command:

```
#/usr/NetFX/bin fccp <options> <src pathname> <dest pathname>
```

For example, to copy a test file from a directory we are in on the HIO system in our System structure to a directory on the GIO system, we would use a command like the following:

```
# /usr/NetFX/bin/fccp -vtD testfile pci_for_network/jbod_xfs
```

See the `fccp` and `fccpd` man pages for more information.

Troubleshooting

There may be problems with your configuration. If you are having problems, take a look at the following Troubleshooting Guides:

Troubleshooting Guideline for Storage

1. Are the SGI patches current and correct?

   ```
   # versions | grep patch
   ```

   See the current README FIRST shipped with your software, or check the Prisa Tech Support Web Page at **www.prisa.com/support**

2. Is correct software loaded?

   ```
   # versions | grep NetFX
   ```

NetFX	Base Environment
NetFX_SCDisk	SCSI Disk Driver
NetFX_scsi	FibreChannel SCSI (SCSI Protocol)
NetFX_Switch	Switch Software

   ```
   # versions | grep xlv
   ```

xlvgdisk	xlv Adaptation Driver (Only needed for 5.3, 6.2, and 6.3)

Are fsd, xlv, and xlvgdisk chkconfig'd on?

3. Does npname in /etc/config/netfx.options match npname in /etc/NLPorts?

4. Is correct adapter information in /etc/config/netfx.options?
   adapter type, slot, position, port
   enable=yes, si=yes
   Switch options (if connecting to a Switch)

   Information is in /var/adm/SYSLOG

5. Does npname in /etc/config/fsd.options match npname in /etc/NLPorts?

6. Is each disk's loop ID correct in /etc/NLPorts?

7. Is each disk's loop ID unique in /etc/NLPorts?

8. Do the adapter and disk(s) have the same channel ID in /etc/NLPorts?

9. Is the WWN correct for each disk in /etc/NLPorts?

Troubleshooting Guideline - Network

1. Are the SGI patches current and correct?
   `# versions | grep patch`
   See the current README FIRST shipped with your software, or check the Prisa Tech Support Web Page at **www.prisa.com/support**

2. Is correct software loaded? `# versions –b | grep NetFX`

NetFX	Base Environment
NetFX_IP	Fibre Channel IP
NetFX_ZIP	IP Accelerator
NetFX_Trans	Transporter
NetFX_Switch	Switch Software

3. Is network chkconfig'd on? `# chkconfig | grep network`
   Is netfx chkconfig'd on? `# chkconfig | grep netfx`

4. Does npname in /etc/config/netfx.options match npname in /etc/NLPorts?

5. Does Fibre Channel IP hostname in /etc/hosts match npname in /etc/NLPorts?

6. Is correct adapter information in /etc/config/netfx.options?
   adapter type, slot, position, port
   enable=yes, ip=yes (IP), tpt=yes (Transporter)
   ZIP (FastIP) option (if using ZIP)
   Switch options (if connecting to a Switch)

   Information is in /var/adm/SYSLOG

7. Is each adapter's loop ID unique in /etc/NLPorts?

8. Do all devices on a same loop have the same channel ID?

9. Is the WWN for each npname correct in /etc/NLPorts?

Appendix A About NetFX Software

Overview

This chapter introduces and describes NetFX storage, network, development, and application software programs. NetFX software supports standard protocols (such as IP and SCSI) as well as sophisticated programs (such as Transporter, FCCP). NetFX software significantly enhances Silicon Graphics IRIX operating system and NetFX adapter performance. The software descriptions include references to manual pages and other documentation.

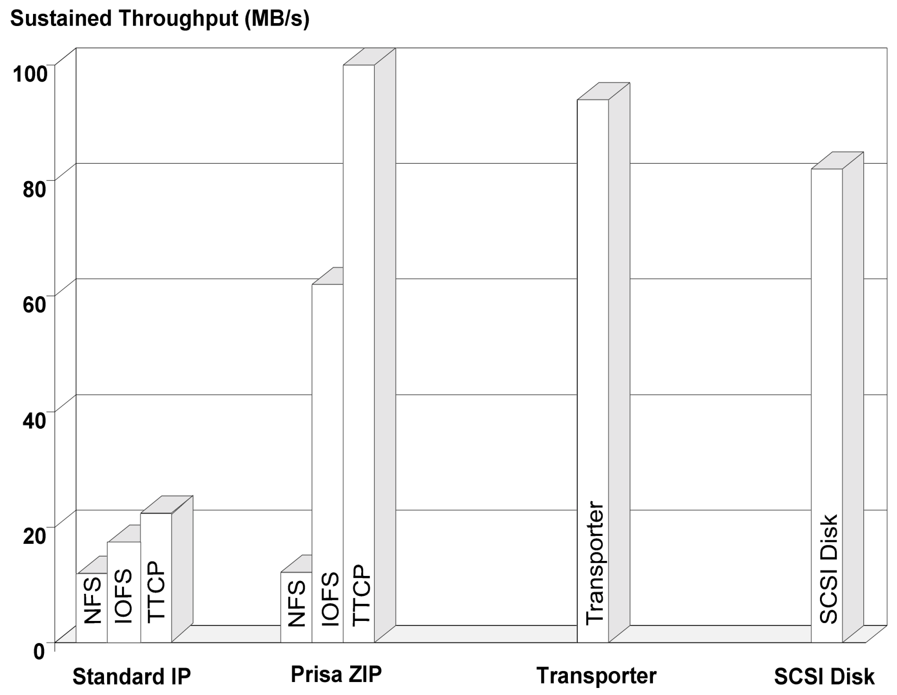
This chapter includes these sections:

Prisa NetFX Software Enhances System Performance

Prisa supplies software with the NetFX adapter board that supports standard Fibre Channel transport services. The software is optimized to sustain high data throughput for block data transfers while supporting high transaction rates for small-message traffic. NetFX supports an average sustained rate of up to 100 megabyte/sec (800 megabit/sec) for multi-megabyte data transfers.

Figure 7-1 NetFX Performance illustrates NetFX software performance gains. (Actual adapter performance depends on the system as well as the specific mix of large block transfer and small-message traffic, which trade off against each other.)

Figure 6-1 NetFX Performance



Prisa NetFX Software

NetFX Fibre Channel Adapter Software

NetFX Base

NetFX Fibre Channel Base software contains the base level hardware drivers and diagnostic utility for the NetFX adapter. It supports point-to-point and arbitrated loop Fibre Channel topologies.

NetFX Switch

Enhancements to the NetFX Fibre Channel Switch software package supports switched topologies.

NetFX Storage Software

NetFX SCSI Disk Driver

The SCSI Initiator allows the host system to use Fibre Channel-connected SCSI devices such as RAID drives and JBODs.

NetFX SCSI (SCSI Initiator)

The NetFX Fibre Channel SCSI software supports SCSI-3 encapsulation over Fibre Channel. It complies with SCSI's FCP standard and uses the SCSI-3 architectural model. NetFX Fibre Channel SCSI is packaged with the SCSI API, which provides a programmatic interface to Fibre Channel SCSI devices. This API is extremely easy to use and implements the standard SCSI-3 command set. For more information, contact Prisa Technical Support.

XLV Generic Disk Adaptation Driver (xlvgdisk)

The disk driver and network software lets the operating system communicate with attached Fibre Channel devices. NetFX disk drivers support Fibre Channel RAID and native Fibre Channel disk drives. These drivers provide full Fibre Channel SCSI capability and are IRIX, XFS, EFS, and XLV compliant.

This package is not required for IRIX release 6.4.

NetFX Network Software

NetFX Fibre Channel IP

This interface supports standard IP and works similar to a standard Ethernet® network card. It supports TCP, UDP, SNMP, ICMP, ARP, and raw socket protocols, allowing standard IP utilities and programs (such as FTP, Telenet, and NFS) to realize Fibre Technology benefits. The SGI IRIX raw socket Snoop and Drain protocols are also supported. For more information, see the section *Defining IP as the communications protocol* (see page 6-29).

NetFX IP Accelerator (ZIP)

NetFX IP Accelerator increases IP transfer rates up to 100 MB/sec and is particularly effective when used to transfer files between systems.

NetFX Transporter

Transporter is a proprietary Prisa protocol that treats transfers between computer systems as solicited memory-to-memory I/O operations, which decreases intermediate copying. This program provides the fastest way possible to transfer large blocks of data between computers. For more information, see *Appendix A Programmer's Guide.*

NetFX Development Software

NetFX Development Environment (Transporter API)

The Transporter API is a programmatic interface to the Transporter protocol. For more information, see *Appendix A Programmer's Guide.*

NetFX Application Software

NetFX Diagnostics

This menu-driven utility guides you through a series of tests to verify NetFX adapters are installed correctly and can send and receive a recognizable Fibre Channel data stream. This program, which performs board and channel level tests, is part of the NetFX Base software package. For more information, see *Chapter 5 Running Diagnostic Tests.*

NetFX Fibre Channel Copy (FCCP)

NetFX Fibre Channel Copy is a utility that uses the Prisa NetFX Transporter protocol to transfer files between devices via Fibre Channel. It has a simple syntax similar to the standard UNIX utility `rcp` and is part of the NetFX Transporter software package. For more information, see the manual pages `fccp` and `fccpd`.

Prisa NetFX Software Hierarchy

The figures below illustrate the NetFX software and utility software hierarchies.

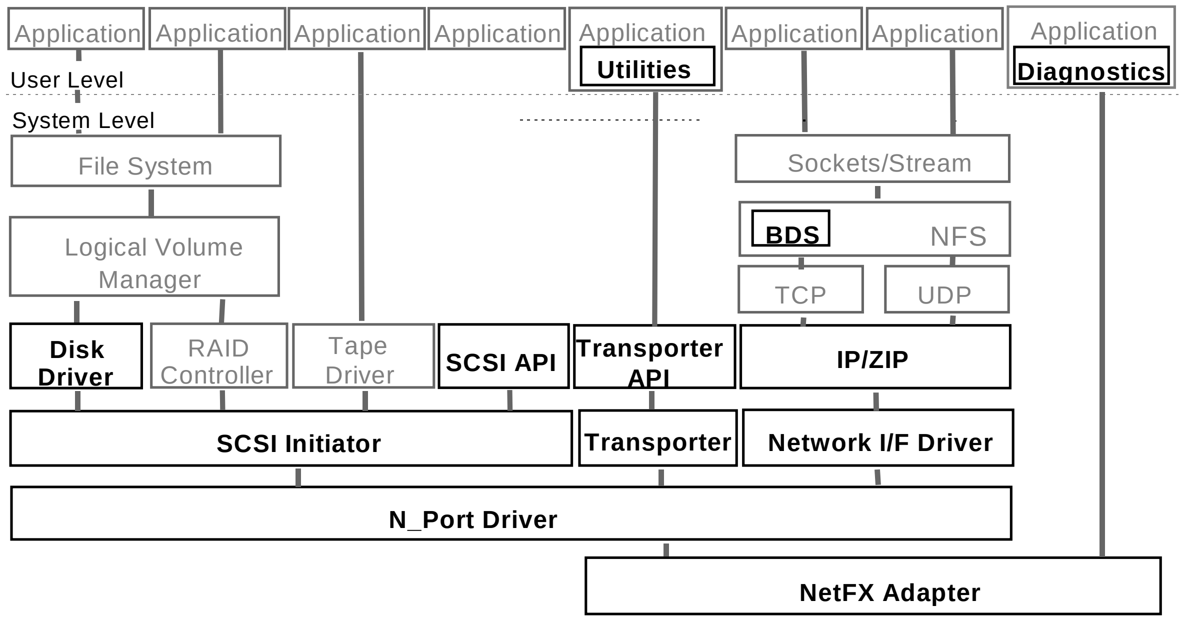


Figure 6-2 NetFX Software Hierarchy

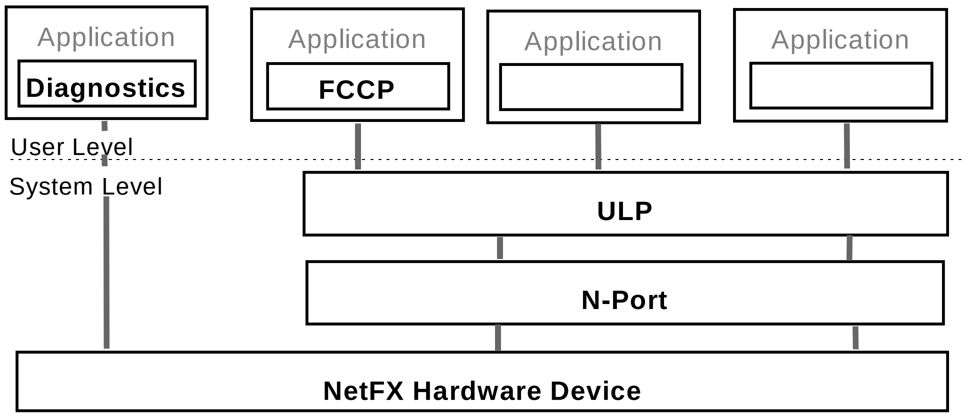


Figure 6-3 NetFX Utility Program Hierarchy

This page intentionally left blank.

Appendix B Programmer's Guide

Overview

This chapter contains information useful for software developers. In this appendix, *italic* text identifies comments provided for reference purposes.

This chapter includes these sections:

SCSI Initiator API

The SCSI Initiator API provides a platform for interfacing to SCSI device drivers for peripherals such as RAID drives, disk drives, tape drives, and plotters. It complies with FCSI's FCP standard, using the SCSI3 architecture model.

This API is extremely easy to use. It implements the standard SCSI3 command set, as described in the following sections. Contact Prisa regarding use of this API.

Transporter™ API

The following is an overview of Prisa's Transporter™ interface. This is intended for applications programmers who need access to a high-speed, large block size transfer protocol. It is assumed that the reader is familiar with Fibre Channel concepts and constructs such as a N_ports, sequences, and World Wide Number identifiers. An understanding of the SCSI protocol is helpful, as is a background in Remote Procedure Call (RPC) programming.

Transporter Overview

In a traditional computer network, data transfer between two entities is typically "unsolicited." The recipient of data does not know ahead of time when packets will arrive, and therefore cannot set up for the transfer and obtain resources to optimize the operation.

The Transporter interface treats data transfers between computer systems as solicited memory-to-memory I/O operations, rather than as conventional unsolicited network transfers. The Transporter paradigm is modeled after the SCSI protocol and allows Transporter to optimize large block transfers between systems. The protocol is broken up into three distinct phases. In the first phase, the *request phase*, a client sends a request to a server initiating an operation. This phase provides the server with enough information to obtain any necessary resources to carry out the request.

The second phase is the *data transfer phase*. On a read, the client requests a certain amount of data from the server, and waits for the server to initiate the transfer out of the data. On a write, the client informs the server of the write request, and waits for the server to initiate the transfer in.

The last phase is the *status (or response) phase*. The server replies to the client's original request with an indication of the status of the operation.

All three phases are explicitly broken out into separate interfaces on the server side. This allows servers more control over the handling of requests and lends itself quite nicely to an RPC-like request loop paradigm. On the client side there is no knowledge of the three distinct phases. Clients make simple read and write requests that encompass all three phases.

Data Formats

The Transporter API places no limitations on the format of the request (or response) from client to server. Developers are free to use any format as long as it can be represented by a (void *) pointer and a length in bytes. Typically, a server will define a well-known interface structure that clients must adhere to. This interface structure contains a command code, and request-specific data.

When transferring data, Transporter requires that both the data and the Transporter length each be 32-bit word aligned.

Transporter Status Codes

All Transporter routines return –1 on error and set the global "errno" to an appropriate value. There are also ancillary return codes for many operations that provide additional status information about the transfer.

Applications that use ptcRead(), ptcWrite(), ptcNoTransfer(), ptsRespond(), ptsTransferIn(), and ptsTransferOut() must check the Transporter status code "ptResult" after each function call (in addition to the normal return code) to determine the true status of the operation. "ptResult" is an enum with these values:

```
ptUndefined = –1,   /* no result yet */
ptSucceeded = 0,    /* operation succeeded */
ptInaccessible,     /* remote device is inaccessible */
ptTimeout,          /* command timed out */
ptProtocolError,    /* transport protocol error detected */
ptLoggedOut,        /* connection to device failed */
ptTransferError,    /* error during data transfer */
ptCanceled,         /* cancelled by user request */
ptNoTransfer,       /* data transfer failed to occur*/
ptRequestError,     /* request was malformed/invalid */
ptMemError,         /* invalid address/parity error */
ptNoMem,            /* insufficient resources for request */
```

Implementation Notes

The interface between an applications program and the kernel is via a device driver. To eliminate a cumbersome ioctl(2) style interface, wrappers for the driver exist. Programmers can access all of these functions via the "libpt.so" and "libpt.a" libraries.

There are two drivers: one for the client (/dev/aptc) and one for the server (/dev/apts). To use any Transporter functionality, these drivers must exist and the appropriate device drivers must be linked into the kernel. This is normally taken care of during installation of the Transporter.

API Conventions

There are two sets of interfaces: one for the client (with the 'ptc' prefix) and one for the server (with the 'pts' prefix). In the Transporter paradigm the "server" is the application program that provides one or more specific types of "service" to clients. A "client" is an applications program that makes requests of a server.

In the interface descriptions, an '(I)' indicates an input parameter, and an '(O)' indicates an output parameter. All functions are callable from either C or C++.

To use the client functions, include this file: `include <ptClient.h>`

To use the server interfaces, include this file: `include <ptServer.h>`

Server API Summary

`ptsOpen`	Establish a Transporter server endpoint.
`ptsClose`	Close a server Transporter endpoint.
`ptsRequest`	Get a Transporter request.
`ptsRespond`	Respond to a Transporter request.
`ptsGetSource`	Get the WWN and port ID corresponding to a request.
`ptsTransferIn`	Initiate an input data transfer.
`tsTransferOut`	Initiate an output data transfer.

Client API Summary

`ptcOpen`	Open a Transporter client endpoint.
`ptcClose`	Close a client Transporter endpoint.
`ptcRead`	Request a read of some number of bytes from a server.
`ptcWrite`	Request a write of some number of bytes to a server.
`ptcNoTransfer`	Send a non-data transfer request to a server.

ptsOpen

Purpose Establish a Transporter server endpoint.

Synopsis

```
int
ptsOpen(
    fc_NameIdentifier nport,
        ptPortID   service_portnum,
        int        max_reqs)
```

Description Before an application can use the server Transporter API, it must identify itself as a provider of services to the interface. Because a given server may provide a variety of services, a specific service is identified by an <nport, service_portnum> tuple.

The server is identified by an fc_NameIdentifier, the World Wide Number (WWN) of an N_port. Clients attempt to connect with the server using the WWN and the server's port number, both of which must be made available to clients via a mechanism external to the Transporter interface.

Parameters `fc_NameIdentifier (I):    server`

The WWN identifying the server.

`ptPortID (I):    service_portnum`

The port number identifying the service.

```
int (I):         max_req
```

The maximum number of requests the server will have on this port number at any one time.

Returns Upon success, ptsOpen returns a non-negative integer (greater than or equal to zero) that is used as a file descriptor in subsequent server Transporter function calls. On error, –1 is returned, and the global errno is set to one of these values:

[EFAULT]	The arguments can't be copied into the kernel.
[EINVAL]	The maximum number of requests is greater than the Transporter limit, ptc_Max_MaxRequests or equals 0.
[ENODEV]	Can't open the Transporter device.
[ENOMEM]	Unable to allocate memory for an internal request structure.

Notes The file /dev/apts must exist and be a character device special file.

See also `ptsClose`

ptsClose

Purpose Close a server Transporter endpoint.

Synopsis

```
  int
          ptsClose(
             int  filedesc)
```

Description ptsClose terminates an application's use of a Transporter server endpoint. Subsequent uses of the file descriptor are invalid. Any outstanding requests that have been received but not yet processed or requests that have been received and are being processed are canceled.

Parameters

```
int (I):     filedesc
```

Transporter server file descriptor.

Returns Upon success, 0 is returned. On error, –1 is returned, and the global 'errno' is set to one of the following values:

[EBADF]	filedesc is not valid.
[EINTR]	A signal was caught during the system call.

See also `ptsOpen`

ptsRequest

Purpose Get a Transporter request.

Synopsis

```
int
```

```
ptsRequest(
     const int      filedesc,
     void           request,
     const u_int    buffer_bytes,
     u_int          *request_bytes,
     ptRequestID    *req_id)
```

Description The first phase of a Transporter server operation is the command phase. A server uses ptsRequest() to retrieve commands from clients. Each command will have a unique tag, 'req_id', that must be used to identify the operation in subsequent Transporter function calls.

Parameters

`const int (I):       filedesc,`

Transporter server file descriptor

`void * (I):          request`

Address of request buffer.

`u_int (O):           request_bytes`

Size of request buffer.

`u_int (O):           request_bytes`

Number of bytes actually returned.

`PtsRequestID * (O):  req_id)`

Request identifier.

Returns On success, 0 is returned, 'request_bytes' is updated to the actual number of bytes in the request, and 'req_id' is set to a unique request ID. On error, -1 is returned, and the global 'errno' is set to one of the following values:

[EFAULT]	Request couldn't be copied into or out of kernel space. Request buffer or request length may be invalid.
[EINTR]	Operation was interrupted.

See also `ptsRespond`

ptsRespond

Purpose Respond to a Transporter request.

Synopsis

```
int
ptsRespond(
       const int     filedesc,
       ptsRequestID  req_id,
       void          *response,
       int           response_bytes,
       u_int          timeout,
       ptResult       *pts_result)
```

Description The last phase of a Transporter operation is the response phase. ptsRespond() provides the server with a mechanism to report the status of an operation back to the client.

Parameters `const int (I):     filedesc`

Transporter server file descriptor.

`ptsRequestID (I):  req_id`

Request ID.

`void * (I):        response`

Pointer to response to send.

`int (I):           response_bytes`

Length of response in bytes.

`u_int (I           timeout`

Currently ignored.

`ptResult * (O):    pts_result`

Result of operation.

Returns Upon success, ptsRespond returns 0, and sets 'pts_result' to one of the Transporter status codes. On error, -1 is returned, and the global 'errno' is set to one of these values:

[EFAULT]	Couldn't copy request into or out of kernel space. Response buffer or response length may be invalid.
[EINVAL]	The size of the response is invalid (0 or greater than pts_Max_RequestBytes).
[EINTR]	Operation was interrupted.
[ENOENT]	The request ID is invalid.

See also `ptsRequest`

ptsGetSource

Purpose Get the WWN and port ID corresponding to a request.

Synopsis

```
int

ptsGetSource(

        const int            filedesc,

        ptsRequestID         req_id,

        fc_NameIdentifier    *source,

        ptPortID             *port)
```

Description A Transporter server may be handling many different requests at any given time. Each request is identified by a request ID. If a server needs more information about the client that initiated the request, it can use ptsGetsource() to obtain the World Wide Number and the port number of the client.

Parameters `const int (I):      filedesc`

Transporter server file descriptor.

`ptsRequestID (I):   req_id`

Request ID to be queried.

Client source port ID.

Client source name ID.

`ptPortID * (O):     port`

Returns On success, ptsGetSource returns 0. On error, it returns -1 and sets the global 'errno' to one of the following values:

[EFAULT]	Request could not be copied into or out of kernel space. 'source' or 'port' may be invalid.
[ENOENT]	The request ID is invalid.

See also `ptsRequest`

ptsTransferIn

Purpose Initiate an input data transfer.

Synopsis

```
int
ptsTransferIn(
        const int       filedesc,
        ptsRequestID    req_id,
        void            *buffer,
        size_t          nbytes,
        u_int           timeout,
        ptResult        *pts_result)
```

Description When a client needs to write data to the server, it uses ptcWrite(). The server retrieves the request via ptsRequest(), obtains the necessary resources, and starts the transfer via ptsTransferIn().

The data will be transferred directly (via DMA) from the Fibre Channel network to the user's buffer pointed to by 'buffer'. The data buffer must be 32-bit word aligned, and the length must also be 32-bit word aligned.

Parameters `const int (I):     filedesc`

Transporter server file descriptor.

`ptsRequestID (I):  req_id`

Request ID.

```
void * (I):        buffer
```

Buffer to transfer data into.

```
size_t (I):        nbytes
```

Number of bytes to transfer.

```
u_int (I):         timeout
```

Timeout time in milliseconds.

```
ptResult * (O):    pts_result
```

Result of operation.

Returns Upon success, the number of bytes actually transferred is returned, 'pts_result' is set to one of the Transporter status codes. On error, –1 is returned and the global 'errno' is set to one of these values:

[EFAULT]	Request could not be copied into or out of kernel space. Data buffer or length may be invalid.
[EINVAL]	The size of the buffer is invalid (0 or greater than pts_Max_DataBytes).
[EINVAL]	The data buffer or number of bytes to transfer is not word-aligned.
[EINTR]	Operation was interrupted.
[ENOENT]	The request ID is invalid.

See also `ptsTransferOut`

ptsTransferOut

Purpose Initiate an output data transfer. .

Synopsis

```
int
   ptsTransferOut(
       const int        filedesc,
       ptsRequestID     req_id,
       void             *buffer,
       size_t           nbytes,
       u_int            timeout,
       ptResult         *pts_result
```

Description When a client needs to read data from the server, it uses ptcRead(). The server retrieves the request via ptsRequest(), obtains the necessary resources, and starts the transfer via ptsTransferOut(). The data will be transferred directly (via DMA) from the server's buffer to the network. The data buffer must be 32-bit word aligned, and the length must also be 32-bit word aligned.

Parameters

`const int (I):      filedesc`

Transporter server file descriptor.

`ptsRequestID (I):   req_id`

Request ID.

`void * (I):         buffer`

Buffer to transfer data from.

`size_t (I):        nbytes`

Number of bytes to transfer.

`u_int (I):         timeout`

Currently ignored.

`ptResult * (O):    pts_result`

Result of operation.

Returns

Upon success, the number of bytes actually transferred is returned, and 'pts_result' is set to one of the Transporter status code. On error, -1 is returned and the global 'errno' is set to one of the following values:

[EFAULT]	Request could not be copied into or out of kernel space. Data buffer or length may be invalid.
[EINVAL]	The size of the buffer is invalid (0 or greater than pts_Max_DataBytes).
[EINVAL]	The data buffer or number of bytes to transfer is not word-aligned.
[EINTR]	Operation was interrupted.
[ENOENT]	The request ID is invalid.

See also `ptsTransferIn`

ptcOpen

Purpose Open a Transporter client endpoint.

Synopsis

```
int
ptcOpen(
    fc_NameIdentifier  server,
    ptPortID           server_port,
    int                max_reqs)
```

Description

Before an application can use the client Transporter API, it must identify itself to the interface. This is accomplished with the ptcOpen() call. The client must specify the identity of the server application, the port number of that server, and the maximum number of outstanding requests that the client can generate.

The server is identified by an fc_NameIdentifier, the World Wide

Number (WWN) of a remote N_port. The client must know this identifier via some mechanism external to the Transporter interface. Likewise, the client and server must agree ahead of time on the value of the server port number. Because servers can provide more than one type of service, each service is identified by a unique port number.

Parameters

`fc_NameIdentifier (I):   server`

The WWN identifying the server.

`ptPortID (I):           server_port`

The port number of the service.

`int (I):    max_req`

The maximum number of outstanding requests the client will be allowed to have at any one time.

Returns

Upon success, ptcOpen returns a non-negative integer (greater than or equal to zero) that is used as a file descriptor in subsequent client Transporter function calls. On error, –1 is returned, and the global 'errno' is set to one of the following values:

[EFAULT]	The arguments can't be copied into the kernel.
[EINVAL]	The max number of requests is greater than the Transporter limit, ptc_Max_MaxRequests or equals 0.
[ENODEV]	Can't open the Transporter device.
[ENOME]	Unable to allocate memory for an internal request structure.

Notes

Upon success, ptcOpen returns a non-negative integer.

The file /dev/aptc must exist and be a character device special file.

See also

`ptcClose`

ptcClose

Purpose

Close a client Transporter endpoint. .

Synopsis

```
int
ptcClose(
    int filedesc)
```

Description

`ptcClose` terminates an application's use of a ptc endpoint. Subsequent uses of the file descriptor are invalid.

Parameters

`int (I):          filedesc`

Client Transporter file descriptor.

Returns

Upon success, 0 is returned. On error, -1 is returned, and the global 'errno' is set to one of the following values:

[EBADF]	filedesc is not valid.
[EINTR]	A signal was caught during the system call.

See also `ptcOpen`

ptcRead

Purpose Request a read of some number of bytes from a Transporter server.

Synopsis

```
int
ptcRead(
      int        fd,
      void       *req,
      u_int      req_bytes,
      u_int      timeout,
      void       *data,
      size_t     data_bytes,
      void       *resp,
      u_int      *resp_bytes,
      ptResult   *presult)
```

Description A client uses pctread() to read data over the Transporter interface. The data will be transferred directly into the buffer indicated by 'data' via DMA, and will be of length 'data_bytes'. The data buffer must be 32-bit word aligned, and the length must be 32-bit word aligned.

All three phases (command, data transfer, and status response) are encapsulated in ptcRead (). The format of the read request pointed to by 'req' is not defined by the interface, but will typically be a structure containing information about the location and length of data to be read. The interface also does not define the format of the response pointed to by 'resp'.

Once the server receives the request, it will obtain the resources necessary for the operation, initiate the data transfer, and fill in the response pointed to by 'resp' to reflect the status of the transfer.

Parameters

`int (I):          fd`

Client Transporter file descriptor.

`void * (I):   req`

Pointer to request describing read operation.

`u_int (I):    req_bytes`

Number of bytes in request 'req'.

`u_int (I):    timeout`

The number of milliseconds to wait for a server response.

`void * (I):   data`

Pointer to data buffer to receive incoming data.

`size_t (I):   data_bytes`

Total length of data buffer.

`void * (I):   resp`

Pointer to response.

`u_int (I/O):  resp_bytes`

On input, it contains the number of bytes in the response. On out put, it contains the actual length of the response.

`ptrResult * (O):         pts_result`

Status of operation.

Returns On success, the number of bytes actually transferred is returned. 'resp_bytes' is updated to the actual number of bytes in the response structure. Additionally, 'presult' is set to one of the Transporter status codes. On error, -1 is returned and the global 'errno' is set to one of the following values:

[EFAULT]	Request could not be copied into or out of kernel space.
[EFAULT]	The user buffer is invalid.
[EINVAL]	The number of bytes in the request structure exceeds the maximum request size, ptc_Max_RequestBytes.
[EINVAL]	The number of bytes to be read exceeds the maximum transfer size, ptc_Max_DataBytes.
[EINVAL]	The data buffer or number of bytes to transfer is not word-aligned.
[ENOSPC]	The Transporter couldn't allocate request structure.

Notes To determine if a transfer completed successfully, there are a number of checks that the client should perform. In the following example, the client wants to read 'nbytes' of data into 'buf'.

```
struct my_request       req;
struct my_response
resp;
u_int           resp_bytes;
ptResult        result;
int             nread;
nread = ptcRead(fd, (void *)&req, sizeof(req), TIMEOUT, buf,
nbytes, (void *)&resp, &resp_bytes, &result);
                          if (nread != -1) {
```

```
                    if (result == ptSucceeded) {
                    if (nread == nbytes)
          /*
*Successful transfer. Could also check that *
*‘resp_bytes’
                   *is a sane value
                   */
          else
                   /* Short read */
      } else
      /*Transporter error, check status code */
  } else {
      /* Error in setting up request */
  )
```

See also ptcWrite, ptcNoTransfer

ptcWrite

Purpose Request a write of some number of bytes to a Transporter server. .

Synopsis

```
int
ptcWrite(
int             fd,
void            *req,
u_int              req_bytes,
u_int           timeout,
void            *data,
size_t          data_bytes,
void            *resp,
u_int              *resp_bytes,
ptResult           *presult)
```

Description A client uses ptcWrite() to write data over the Transporter interface.

The data will be transferred directly from the buffer indicated by ‘data’ via DMA, and must be of length ‘data_bytes’. The data buffer must be 32-bit word aligned, and the length must also be 32-bit word aligned.

All three phases (command, data transfer, and status response) are encapsulated in ptcWrite(). The format of the write request pointed to by ‘req’ is not defined by the interface, but will typically be a structure containing information about the location and length of data to be written. The interface also does not define the format of the response pointed to by 'resp'.

Once the server receives the request, it will obtain the resources necessary for the operation, initiate the data transfer, and fill in the response pointed to by 'resp' to reflect the status of the transfer.

In the example below, *italic* text identifies comments for reference purposes.

Parameters

`int (I):        fd`

Client Transporter file descriptor.

`void * (I):     req`

Pointer to request to send.

`u_int (I):      req_bytes`

Number of bytes in request.

`u_int (I):      timeout`

The number of milliseconds to wait for a response from the server.

`void * (I):     data`

Pointer to data buffer to write.

`size_t (I):     data_bytes`

Total length of data buffer.

`void * (I):     resp`

Pointer to response.

`u_int (I/O):    resp_bytes`

On input, it contains the number of bytes in the response. On output, it contains the actual length of the response.

`ptResult* (O):` presult

Status of operation.

Returns

On success, the number of bytes actually transferred is returned. 'resp_bytes' is updated to the actual number of bytes in the response structure. Additionally, 'presult' is set to one of the Transporter status codes. On error, -1 is returned and the global 'errno' is set to one of the following values:

[EFAULT]	Request could not be copied into or out of kernel space.
[EFAULT]	The user buffer is invalid.
[EINVAL]	The number of bytes in the request structure exceeds the maximum request size, ptc_Max_RequestBytes.

[EINVAL]	The number of bytes to be read exceeds the maximum transfer size, ptc_Max_DataBytes.
[EINVAL]	The data buffer or number of bytes to transfer is not word-aligned.
[ENOSPC]	The Transporter could not allocate a request structure.

See also `ptcRead, ptcNoTransfer`

ptcNoTransfer

Purpose Send a non-data transfer request to the server.

Synopsis

```
int
ptcNoTransfer(
    int           fd,
    void          *req,
    u_int         req_bytes,
    u_int         timeout,
    void          *resp,
    u_int         *resp_bytes,
    ptResult      *presult)
```

Description A client uses ptcNoTransfer() to send a command to a server. This command does not involve a data transfer (unlike ptcRead and ptcWrite).

Typically, a client performs some series of operations using ptcNoTransfer() to prepare for a data transfer.

In the example below, italic text identifies comments for reference purposes.

Parameters

```
int (I):      fd
```

Client Transporter file descriptor.

```
void * (I):   req
```

Pointer to request to send.

```
u_int (I):    req_bytes
```

Number of bytes in request.

```
u_int (I):      timeout
```

The number of milliseconds to wait for a response from the server.

```
void * (I):        resp
```

Pointer to response.

```
u_int (I/O):       resp_bytes
```

On output, it contains the actual length of the response.

`ptResult * (O):    presult`

Status of operation.

Returns On success, ptcNoTransfer returns 0, and ptResult is updated with the appropriate Transporter status code. On error, -1 is returned, and the global 'errno' is set to one of these values:

[EFAULT]	Request couldn't be copied into or out of kernel space.
[EINVAL]	The request structure exceeds the maximum number of bytes, ptc_Max_RequestBytes.
[ENOSPC]	The Transporter could not allocate a request structure.

See also `ptcRead, ptcWrite`

This page intentionally left blank.

Appendix C Loop ID Reference Table

This table is based on the Fibre Channel Arbitrated Loop Direct Attach SCSI Profile (Private Loop), Version 2.00. Keep these notes in mind as you work with Loop IDs:

- Higher numerical Loop_IDs have high arbitration priority.

  Lowest priority Loop_ID is 0 (zero)

  Highest priority Loop_ID is 126, reserved for FL_Port or N/FL_Port

  Highest priority private NL_Port Loop_ID is 125

- 127 is not a valid Loop_ID, but is reserved to indicate the NL_Port has no preferred address.

AL_PA	LOOP ID		AL_PA	LOOP ID	
(hex)	(hex)	(decimal)	(hex)	(hex)	(decimal)
EF	00	0	CA	13	19
E8	01	1	C9	14	20
E4	02	2	C7	15	21
E2	03	3	C6	16	22
E1	04	4	C5	17	23
E0	05	5	C3	18	24
DC	06	6	BC	19	25
DA	07	7	BA	1A	26
D9	08	8	B9	1B	27
D6	09	9	B6	1C	28
D5	0A	10	B5	1D	29
D4	0B	11	B4	1E	30
D3	0C	12	B3	1F	31
D2	0D	13	B2	20	32
D1	0E	14	B1	21	33
CE	0F	15	AE	22	34
CD	10	16	AD	23	35
CC	11	17	AC	24	36
CB	12	18	AB	25	37
(hex)	(hex)	(decimal)	(hex)	(hex)	(decimal)
AA	26	38	67	48	72
A9	27	39	66	49	73
A7	28	40	65	4A	74
A6	29	41	63	4B	75
A5	2A	42	5C	4C	76
A3	2B	43	5A	4D	77

AL_PA	LOOP ID		AL_PA	LOOP ID	
9F	2C	44	59	4E	78
9E	2D	45	56	4F	79
9D	2E	46	55	50	80
9B	2F	47	54	51	81
98	30	48	53	52	82
97	31	49	52	53	83
90	32	50	51	54	84
8F	33	51	4E	55	85
88	34	52	4D	56	86
84	35	53	4C	57	87
82	36	54	4B	58	88
81	37	55	4A	59	89
80	38	56	49	5A	90
7C	39	57	47	5B	91
7A	3A	58	46	5C	92
79	3B	59	45	5D	93
76	3C	60	43	5E	94
75	3D	61	3C	5F	95
74	3E	62	3A	60	96
73	3F	63	39	61	97
72	40	64	36	62	98
71	41	65	35	63	99
6E	42	66	34	64	100
6D	43	67	33	65	101
6C	44	68	32	66	102
6B	45	69	31	67	103
6A	46	70	2E	68	104
69	47	71	2D	69	105
(hex)	(hex)	(decimal)	(hex)	(hex)	(decimal)
2C	6A	106	1B	75	117
2B	6B	107	18	76	118
2A	6C	108	17	77	119
29	6D	109	10	78	120
27	6E	110	0F	79	121
26	6F	111	08	7A	122
25	70	112	04	7B	123
23	71	113	02	7C	124
1F	72	114	01	7D	125
1E	73	115	00	7E	126
1D	74	116	--	7F	127

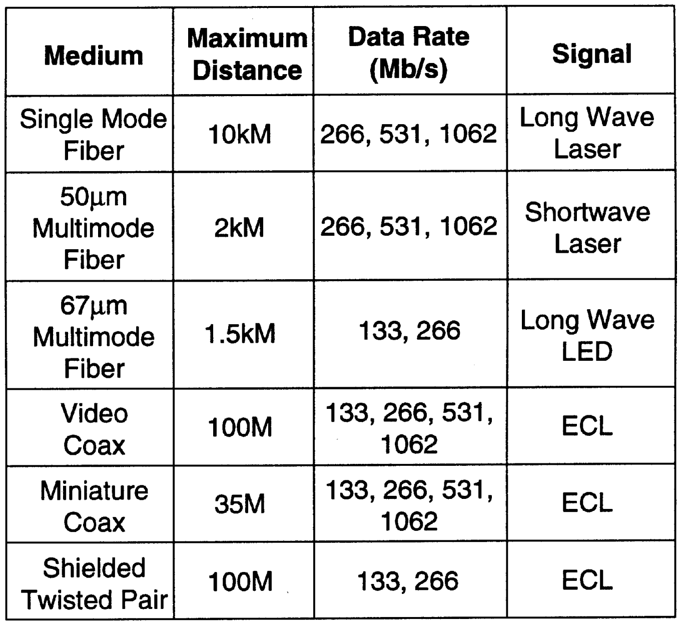
Appendix D The Fibre Channel Hierarchy

The Fibre Channel structure is defined as a multi-layered hierarchy of functional levels. Five layers define the physical media and transmission rates, encoding scheme, framing protocol and flow control, common services and the upper layer application interfaces. FC-0, the lowest layer, specifies the physical features of the media, transmitters, receivers and connectors, including electrical and optical characteristics, transmission rates and other physical elements of the standard. Fibre Channel has been defined to enable system integrators to select several combinations of speed and distance. Within each media class, the components can be selected to suit the economic and performance objective of the application.

FC-1 , defines the 8B/10B encoding/decoding scheme used to integrate the data with the clock information required by serial transmission techniques. Using 10 bits to represent each 8 bits of data, the encoding also provides a first level of data integrity protection.

FC-2 , defines the rules for framing data to be transferred between ports, a look-ahead sliding-window flow control scheme, different mechanism's for circuit and packet switched classes of service, the error detection techniques, and means of managing the sequence of data transfer. All frames in a single transfer are uniquely identified by sequential numbering enabling the receiver to tell not only if a frame is missing, but also which one.

FC-3 , provides common services required for advanced features such as striping and hunt groups (i.e., the ability for more than one port to respond the same alias address).



Medium	Maximum Distance	Data Rate (Mb/s)	Signal
Single Mode Fiber	10kM	266, 531, 1062	Long Wave Laser
50µm Multimode Fiber	2kM	266, 531, 1062	Shortwave Laser
67µm Multimode Fiber	1.5kM	133, 266	Long Wave LED
Video Coax	100M	133, 266, 531, 1062	ECL
Miniature Coax	35M	133, 266, 531, 1062	ECL
Shielded Twisted Pair	100M	133, 266	ECL

Figure C-1 Media Type/Speed

FC-4 provides the seamless integration of existing standards, by accommodating a number of other protocols such as SCSI, TCP/IP, FDDI, HIPPI, ATM, Ethernet, Token Ring and IPI. In all, Fibre Channel permits up to 255 different types of protocols.

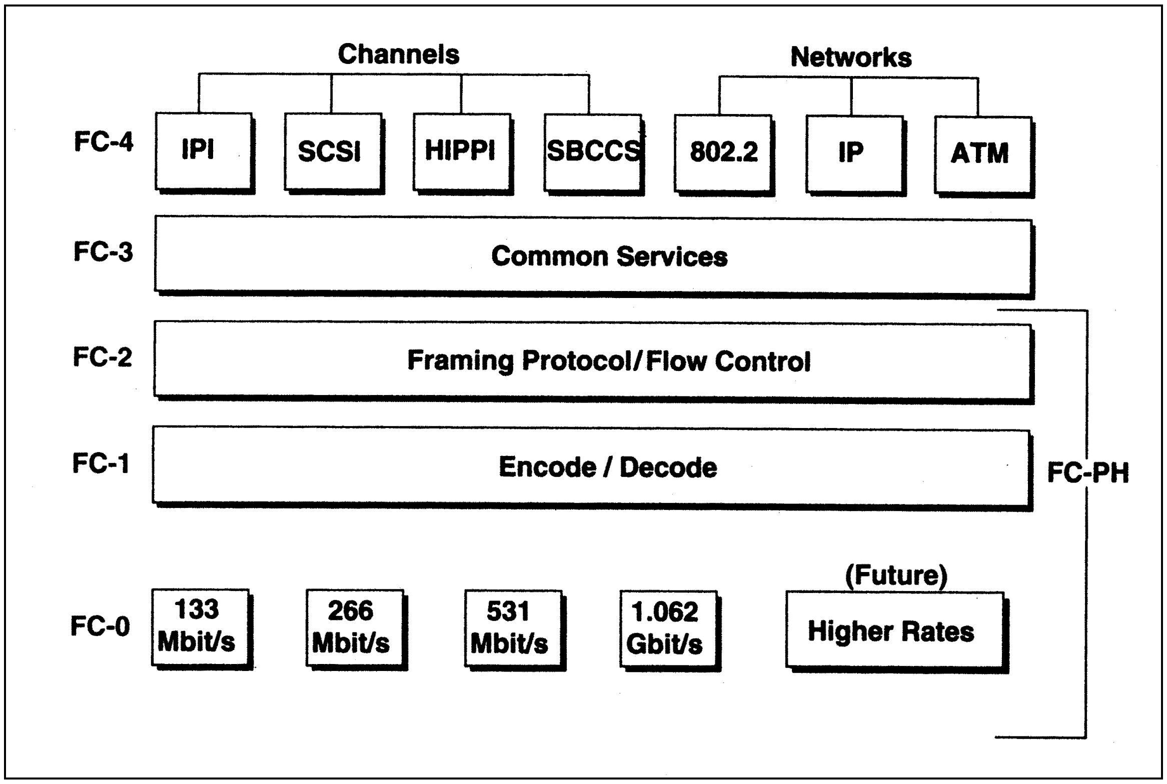


Figure C-2 The Fibre Channel Hierarchy

Fibre Channel combines the best attributes of a channel with those of a network through a simple technique: it provides a means to transfer data between a buffer at the source device (e.g., a RAID drive) and another buffer at the destination device (e.g., a workstation). Fibre Channel ignores the data itself and how is formatted; it simply takes what is in the sending buffer and transports it to the receiving buffer as rapidly as possible. After initial handshaking, control of the rate of data flow is handled by the receiving device indicating the amount of available memory buffer available. Simple error correction is handled in hardware much like a channel. If a data transfer fails due to congestion then a re-try occurs immediately, without consulting system software. More complex error recovery is passed back to the central processor similar to a network environment.

Fibre Channel has established four levels of communication across the links:

SIGNALING occurs via ordered sets, which are four 10-bit characters used for such functions as start-of-frame, end-of-frame, link start-up and special user-defined commands.

FRAMES are the smallest undivided packet of data sent over the connection. Addressing is done with the frame header. The complete frame consists of start-of-frame delimiter, frame header, optional payload header, data payload up to 2048 bytes, 32-bit CRC, and end-of-frame delimiter.

SEQUENCES are composed of one or more related frames flowing in the same direction on a link. They constitute the key unit of transfer between ports which have negotiated available buffers. Each sequence is identified uniquely and each frame within each sequence is individually numbered to facilitate error detection and re-assembly upon arriving at its destination. This level is the recovery boundary, with re-transmission of complete sequences initiated upon error detection.

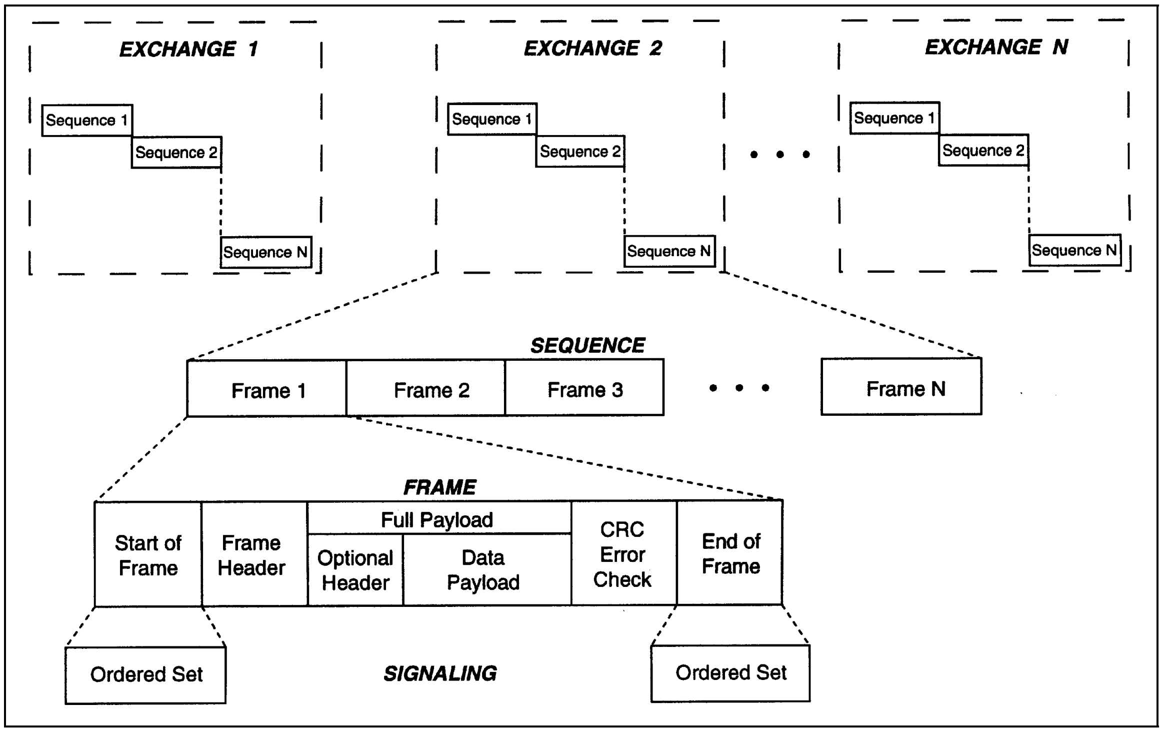


Figure C-3 The Communication Structure

EXCHANGES consist of one or more non-concurrent sequences in a communication series between two devices. Several Exchanges between the same two devices may be occurring at the same time, with each being in a different phase of progress (e.g., initial handshake, data transfer, close of transfer, etc.). Within a single exchange, only a single sequence may be active at any one time, while sequences for different exchanges may be progressing simultaneously.

Flow Control

Key to its high performance is the use of a "fabric" for interconnecting Fibre Channel ports. While primarily oriented towards switched interconnections, the Fibre Channel fabric does define other topologies. The fabric concept is straightforward: It is the responsibility of the Fibre Channel port to manage a simple point-to-point connection with the fabric, while the fabric itself handles all station management associated with routing and error recovery in its network of attachments. Fabric size is not physically constrained, but does have a limit of about 16 million addresses.

Classes of Service

To accommodate a wide range of communication needs, Fibre Channel defines three different classes of service.

CLASS 1, a circuit switched connection, functions much in the same way as today's dedicated physical channels. No other devices can share the engaged link when a Class 1 connection has been established between two devices.

CLASS 2 is a connectionless, frame-switched link which provides guaranteed delivery with acknowledgment of receipt. As with traditional packet-switched systems, the path between two ports is not dedicated, allowing for shared use of the link's bandwidth.

INTERMIX is a hybrid mode between Class 1 and Class 2 services. It reserves the full Fibre Channel bandwidth for a dedicated Class 1 connection, but also allows connectionless Class 2 traffic to share the link on an "as available" basis.

CLASS 3 is also a connectionless "datagram" service that allows data to be sent rapidly to multiple devices attached to the fabric, but no confirmation of receipt is given. By not having to wait for confirmation, Class 3 service speeds the time of transmission. However, if a single-user's link is busy, the system will not know to re-transmit the data.

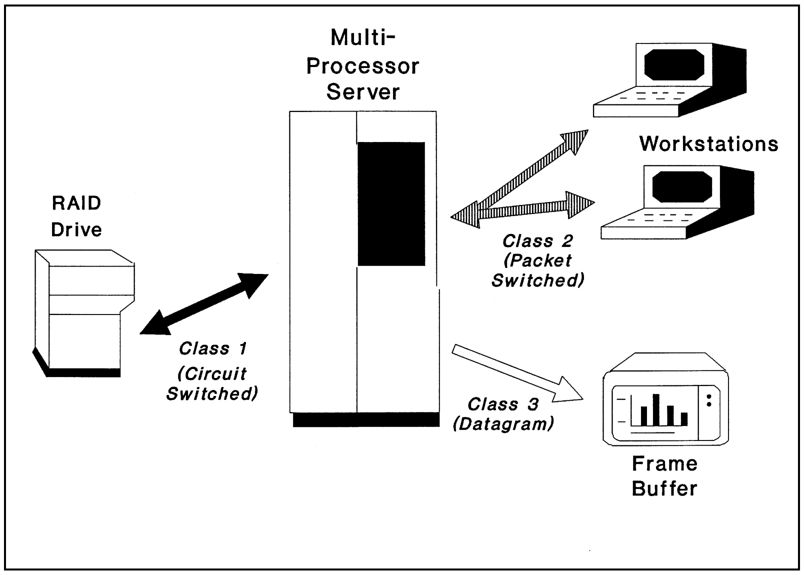


Figure C-4 The Classes of Service

This page intentionally left blank.

Index