Intelligent Peripheral Fault Manager for Tru64 UNIX

Installation and User's Guide

Part Number: AA-QN0FD-TE

June 1999

Operating System and Version: *Tru64 UNIX*, Version 4.0D or 4.0E

Software Version: Version 2.2

Compaq Computer Corporation
Maynard, Massachusetts

February 1998
June 1999

S3634

Table of Contents

4 Understanding IPFM

5 Alarm Utility Operator Interface

6 User Application Program Interface

F Hardware Reference

Glossary

Index

Figures

Tables

Preface

About This Guide

This guide provides installation information and operator instructions for the *Intelligent Peripheral Fault Manager* software.

Note

Your *AlphaServer* system is shipped to your site with the operating system, *Intelligent Peripheral Fault Manager* and other related IP Platform software installed. Keep this guide with your distribution kit. You will need it to install maintenance updates or to reinstall the product for any other reason.

The *Intelligent Peripheral Fault Manager* is a component of the *AlphaServer Intelligent Peripheral Platform* for the *Tru64 UNIX* operating system. It consists of:

- Alarm panel/fault detection hardware and UNIX device driver
- Fault detection software
- Fault database managing software
- An interface to Network Management Stations
- An interface to a user (operator)
- An application program interface (API)

Intended Audience

This document is intended for system managers and programmers trained in software installation, system management and programming who will be using the *Intelligent Peripheral Fault Manager* software.

Structure of This Document

This guide is organized as follows:

Chapter 1, Intelligent Peripheral Fault Manager Overview – Describes new and changed features and provides a high level description of the *IPFM* interfaces.

Chapter 2, Installing IPFM – Provides pre-installation and installation instructions, including how to run the installation verification procedure (IVP).

Chapter 3, Configuring the IPFM Software – Describes the configuration of *DECevent* and the Network Management Station, and the customization of the *IPFM* configuration file.

Chapter 4,Understanding IPFM – Describes the three components of the *IPFM* Event Manager: the Alarm Database Manager (ADM), the SNMP Subagent, and the Alarm Panel Manager. This chapter also discusses the *IPFM* Event Detector and *IPFM* alarm subsystem.

Chapter 5, Alarm Utility Operator Interface – Describes how to access and use the *IPFM* operator interface to set, clear and acknowledge alarms.

Chapter 6, User Application Program Interface – Describes how to use the *IPFM* API commands to enable applications to set, clear or acknowledge alarms.

Appendix A, Sample Installation Script – Provides a log file of a representative *IPFM* installation.

Appendix B, Files Installed on the System – Lists the directory path and names of files installed on the system.

Appendix C, Configuration File – Describes the configuration file used to initialize the Event Manager and Event Detector and provides the commented script.

Appendix D, SNMP Management Information Base – Provides the *IPFM* private MIB in ASN.1 format.

Appendix E, Operation of the aptest Utility – Describes the Alarm Panel Test (aptest) utility, and the tests that can be performed using it.

Appendix F, Hardware Reference – Describes the dry contact terminal connectors and the alarm input wiring of the *AlphaServer Intelligent Peripheral Platform*.

Glossary – Defines technical terms related to the product.

Index – Locates the main topics in this guide.

Related Documentation

For additional information on the *AlphaServer Intelligent Peripheral (IP) Platform* subsystem components and related software, refer to the documentation in the following tables. Order numbers may change as documents are revised or updated. Check with your Compaq sales representative for additional information.

AlphaServer IP Peripheral Platform	Order Number
AlphaServer Intelligent Peripheral Platform System Manager's Guide	AA–QU0JC–TE
AlphaServer Intelligent Peripheral Platform Hardware Owner's Guide	EK–ASIP2–OG

AlphaServer 1000A Processor	Order Number
AlphaServer 1000A Rackmount Owner's Guide	EK–RMNOR–OG
DIGITAL UNIX Installation Guide	AA–QTLGB–TE
StorageWorks KZPSA PCI-to-SCSI Storage Adapter User's Guide	EK–KZPSA–UG

BA35x-Sx Modular Storage Shelf	Order Number
BA350 Modular Storage Shelf Subsystem User's Guide	EK–BA350–UG
BA350 Modular Storage Shelf Subsystem Configuration Guide	EK–BA350–CG
SCSI Signal Converter Service Manual/DWZZB-VA	EK–DWZAA–SV
RZ Series Disk Drive Installation Guide - Models RZ35, RZ26, RZ27, RZ28	EK–DRZ01–IG

Software Documentation	Order Number
ServerWORKS Manager Overview and Installation	ER–4QXAA–UA

Conventions

Conventions Used in This Guide

Convention	Description
IP	IP is an industry-standard acronym for *intelligent peripheral*.
#	A pound sign (#) is the default superuser prompt.
%	A percent sign (%) is the default user prompt.
Ctrl/C	This symbol indicates that you must press the Ctrl key while you simultaneously press another key (in this case, C).
`<RETURN>`	In examples, this symbol indicates that you press the Return key.
% **`cat`**	In interactive examples, typed user input appears in a bold typeface.
`monospaced`	In text, this typeface indicates the exact name of a command, routine, partition, pathname, directory, or file. This typeface is also used in interactive examples and other screen displays.
`UPPERCASE` `lowercase`	The *Tru64 UNIX* operating system differentiates between lowercase and uppercase characters. Literal strings that appear in text, examples, syntax descriptions, and function definitions must be typed exactly as shown.
`[y]`	In a prompt, square brackets indicate that the enclosed item is the default response. For example, `[y]` means the default response is Yes.

1
Intelligent Peripheral Fault Manager Overview

The *Intelligent Peripheral Fault Manager (IPFM)* is the fault management component of the *AlphaServer Intelligent Peripheral (IP) Platform*, which provides a full suite of voice, communications, management, processing, and storage resources that can be accessed and controlled by software running on the platform. The *AlphaServer Intelligent Peripheral Platform* is intended for use in telecommunications networks and environments, supporting real-time communications services and applications. In addition to applications developed by software partners, it also supports general specifications for telecommunications networks such as environmental, network interface, availability requirements, and fault management.

The *IPFM* product detects and reports alarms at severity levels of Critical, Major, or Minor. Alarms are both audible and visible and are displayed on the *IPFM* alarm indicator panel on the top of the equipment rack. These alarms can also be forwarded to a Network Management Station using the SNMP trap mechanism. Interfaces to *IPFM* include the operator interface, SNMP Subagent, application program interface, and alarm panel/fault detection interface. These interfaces are introduced in the following sections.

New and Changed Features Added in V2.1

The following features were added in the V2.1 and V2.1a releases:

- SNMP Trap Filter – Alarms received with a severity below a specified trap threshold will not be forwarded as SNMP traps. This allows the user to configure the type (and quantity) of SNMP traps that get generated. It is possible to use this trap filter to disable SNMP traps from being generated.
- Event Detector polling interval limits – The *IPFM* event detector polling interval is now limited to between 1 second and 5 minutes.
- The *IPFM* alarms now have seconds included in the timestamp.
- Process Monitoring – *IPFM* can be configured to monitor specific processes, and if missing, will run a user-supplied script. It is also possible to specify a count of the number of copies that a given process should have.
- A new API routine IPFM_GET_EVENT has been added. IPFM_GET_EVENT can be used to retrieve alarms as they occur.
- A program linked against the IPFM_USER_API no longer exits automatically if the Alarm Database Manager (ADM) is not running.
- Dry contact relays have been added to the indicator module.

New and Changed Features Added in V2.2

The following features were added in the V2.2 release:

- The maximum number of alarms that *IPFM* can handle has been increased from 100 to 2048.
- The *IPFM* alarm filename convention has been changed. A new file is created daily with the date incorporated in the filename.
- The ipfm_user_api routine has the following new options:
  - CLR_ALL - Clear all alarms
  - ACK_ALL - Acknowledge all alarms
  - REPOST_ALL - Re-post all alarms
  - Get the number of outstanding alarms
  - Get the outstanding alarms
  - Set, Clear and Acknowledge Warning messages
  - Set, Clear and Acknowledge Informational messages
- New MIB variables to allow an NMS to issue a CLR_ALL (clear all alarms), ACK_ALL (acknowledge all alarms), and REPOST_ALL (re-generate SNMP traps for all outstanding alarms).
- The audible alarm can be disabled within the configuration file settings
- Fixed a bug in the temperature threshold environmental monitoring.
- The environmental temperature monitoring will use the dynamic variable ENVMON_HIGH_THRESH as the high temperature threshold (if it has been configured using the UNIX envconfig utility). If this variable has not been set, the read-only “high temperature threshold” from the server system MIB will be used.
- Fixed the process monitoring example in the ipfm.conf configuration file
- Fixed bug with *IPFM* menu. When entering alarm text, either the backspace key or the delete key (depending on the terminal emulator) would cause control characters to be imbedded in the text string making them unprintable. If non ASCII-numeric characters are found in the event text, the entry will be rejected, and the user will be re-prompted for input.

IPFM Operator Interface (User Menu/Display)

The *IPFM* operator interface allows IP system managers to:

- view outstanding alarms
- set/acknowledge/clear alarms
- request information for the local system

When the *IPFM* operator interface is invoked, a screen display appears. The bottom portion of the screen display is used for menu options and command inputs. The top portion contains a list of the outstanding events within the local processor. If this information exceeds a full page, the Next Page and Previous Page keys can be used to access the additional information.

IPFM SNMP Interface

IPFM contains an SNMP Subagent that communicates with Network Management Stations (NMS) such as ServerWORKS or TeMIP for *Tru64 UNIX*. The SNMP Subagent handles SNMP management commands targeted for its private Management Information Base (MIB) and sends out traps to any configured NMS when an alarm condition occurs or is modified.

IPFM Application Program Interface (API)

IPFM provides a defined interface that can be used in developing applications. The interface provides command definitions to set, acknowledge, and clear events in the local system alarm event database.

IPFM Alarm Panel/Fault Detection Interface

IPFM includes an alarm panel, fault detection hardware, and a *Tru64 UNIX* device driver. The alarm panel contains audio and visual alarm indicators for the critical, major, and minor events detected on the local system. Some of these events are detectable by a sensor module mounted in the *AlphaServer IP Platform* ISA expansion chassis. Other events are detectable by the -48V power inverter. In the absence of the -48V power inverter, up to three user-defined events may be defined.

IPFM Event Database

The *IPFM* maintains a database of events occurring on the local system. The database contains environmental events such as the crossing of thermal, fan, and power supply thresholds reported by the *IPFM* fault detection circuits. It also contains events related to thresholds of the local file system and accessibility of disks. Each event is rated as critical, major, minor, or warning.

2
Installing IPFM

The installation guidelines in this chapter are useful when installing the *IPFM* software.

Note

Depending on the level of Factory Installation Services (FIS) you ordered with your system, some or all of the following installation procedures may already have been performed.

Use this document in conjunction with the following related documents:

AlphaServer Intelligent Peripheral Platform Hardware Owner's Guide

AlphaServer Intelligent Peripheral Platform System Manager's Guide

DIGITAL UNIX Installation Guide

Release Notes

The *IPFM* software provides online release notes. Compaq strongly recommends that you read the release notes, because they may contain information about changes to the installation and the use of the product. The release notes are in the following file:

`usr/opt/IPFM/doc/release.notes`

Installation Requirements

The *Tru64 UNIX* `setld` utility is used to install the *IPFM* software. The *Tru64 UNIX* operating system, Dialogic Drivers for *Tru64 UNIX* software, and other required layered software products are installed at the factory. Additionally, the Dialogic software is pre-configured to work with the telephony and voice hardware options selected by the customer.

After installation, set up the *IPFM* software configuration file:

/`usr/opt/IPFM/bin/online/ipfm.conf`

Prerequisites

The following software must be installed prior to installing the *IPFM* product:

- *Tru64 UNIX* Version 4.0D or 4.0E
- *DECevent* V2.6 or greater

See the *AlphaServer Intelligent Peripheral Platform System Manager's Guide* for additional information.

Distribution Kits

Use the bill of materials (BOM) to check the contents of your *IPFM* software and documentation distribution kits. In addition to this guide, the distribution kits include the following items:

- A CD-ROM optical disk
- The *AlphaServer Intelligent Peripheral Platform System Manager's Guide*

If your distribution kits are damaged or incomplete, contact your Compaq representative.

Pre-Installation Information

The following sections describe the information and computing environment you need before installing the *IPFM* software.

Installation Time

Installation of the *IPFM* software should take approximately five minutes. Installation time required for the operating system and required layered products depends on each product. The *IPFM* software consists of the following subsets:

1. IPFMALARM220 - Event Manager and Event Detector processes
2. IPFMAPI220 - *IPFM* application program interface
3. IPFMDRIVER220 - *IPFM* alarm subsystem device driver
4. IPFMTEST220 - Fault Manager tests

Privileges Required for Installation

You must have superuser privileges to install the *IPFM* software.

Disk Space Required

You need 1000 kilobytes of available disk space to install the *IPFM* software. To check total space and free space for the directories where the *IPFM* software files will reside, enter the `df` command. A display similar to the following example is displayed on your screen, showing available free space. This free space must accommodate the subset requirements of the product.

```
Filesystem     512-blocks      Used       Avail    Capacity   Mounted on

/dev/rz0a          126334      99098      14602    87%        /

/dev/rz0g         1732204      671832     887150   43%        /usr
```

Backing Up Your System Disk

Compaq recommends that you back up the system disk before you install any software.

Use the backup procedures established at your site. For details on performing a system disk backup, see your *Tru64 UNIX* system management documentation.

The following steps are required to install the *IPFM* software on your workstation:

1. Boot and configure the operating system.
2. Install the layered product Product Authorization Keys (PAK).
3. Install the layered product software.

Installing the Product Authorization Keys (PAK)

Log on to the system using the `root` account. Using the `lmfsetup` *Tru64 UNIX* script, enter the PAK number (found on your license) for this product. See your *Tru64 UNIX* system administration documentation for information on using the `lmfsetup` script.

Installing the IPFM Software

Installation includes the starting and ending steps that follow, as well as the subsequent configuration and verification procedures. Installation hints are provided in Table 2-1.

Starting the Installation

To install *IPFM*, perform the following steps:

1. Start from the `root` directory.

   ```
   # cd /
   ```

2. If `/mnt` is to be the destination directory, check to see that no files are present before you load the *IPFM* software.

   ```
   # ls /mnt <RETURN>
   ```

   If the directory is empty, the system displays `./ ../.`

   Alternatively, you can create a new directory as follows:

   ```
   # mkdir /nnn <RETURN>>
   ```

   where *nnn* is the name of the new directory.

3. Load the *IPFM* software CD into the CD-ROM device.
4. Mount the CD-ROM device using `/mnt` or the new directory name *nnn* as follows:

   ```
   # mount -r /dev/rz5c /mnt
   ```

   where `/dev/rz5c` is an example name for a CD-ROM device.

   The light on the CD-ROM player flashes. The cursor returns to the screen.

5. To list the files and directories on the CD, enter the following command:

   ```
   # ls /-alF /mnt <RETURN>
   ```

6. To install the *IPFM* software, enter the following command:

   ```
   # setld -l /mnt/IPFM220/kit
   ```

   where the `220` in `IPFM220` indicates version V2.2.

Ending the Installation

At the conclusion of the *IPFM* software installation, follow these steps:

1. Return to the root directory, and enter the following commands:

```
# cd / <RETURN>
# umount /mnt <RETURN>
```

2. When the cursor returns to the screen, remove the CD from the CD-ROM device.

Run Time Configuration File

The *IPFM* configuration file contains the following information:

- Event Category - The category of event to be detected and reported.
- Event Severity - The severity level of the event.
- Event Text - The text of the event.

The *IPFM* configuration file is located at `/usr/opt/IPFM/bin/online/ipfm.conf`

The user can modify the *IPFM* configuration file to:

- Enable or disable the detection of events
- Change the severity of an event
- Change the text used for an event
- Set SNMP trap thresholds
- Perform process monitoring
- Disable the audible alarm

Note

The event severity and event text cannot be modified on the “system” detected events. The severity and text are determined from the component generating the error.

Appendix C contains the default *IPFM* configuration file.

Running the Installation Verification Procedure (IVP)

To run the IVP for a subset, enter the following information at the command prompt:

```
# /usr/sbin/setld subset_name <RETURN>
```

Where *subset_name* is the name of the subset to be verified (`Alarm`, `Menu` or `Drv`).

The following three examples, one for each subset, illustrate the installation verification procedure.

IVP Example 1: IPFM Event Manager/Detector Files

```
# /usr/sbin/setld -v IPFMALARM220


IPFMALARM - Event Man/Det Process (IPFMALARM220)
        0 verification errors encountered.
        0 corrections performed.

IVP, information: IPFMALARM220 is properly installed.
```

IVP Example 2: IPFM Application Interface Files

```
# /usr/sbin/setld -v IPFMAPI220


IPFMAPI - IPFM Application Interface (IPFMAPI220)
        0 verification errors encountered.
        0 corrections performed.

IVP, information: IPFMAPI220 is properly installed.
```

IVP Example 3: IP Device Driver Files

```
# /usr/sbin/setld -v IPFMDRIVER220


IPFMDRIVER - Alarm Panel Driver (IPFMDRIVER220)
        0 verification errors encountered.
        0 corrections performed.

IVP, information: IPFMDRIVER220 is properly installed.
```

IVP Example 4: IPFM Test Files

```
# /usr/sbin/setld -v IPFMTEST220


IPFMTEST - IPFM Fault Manager Tests (IPFMTEST220)
        0 verification errors encountered.
        0 corrections performed.

IVP, information: IPFMTEST220 is properly installed.
```

If an installation fails the installation verification procedure, the executable files contained in `/usr/opt/IPFM` will be removed automatically.

Installation Hints

Table 2-1 provides hints on how to accomplish various installation tasks.

Table 2-1: Installation Hints

If you want to ...	Then, take this action
Exit the installation	Because you are not satisfied with the backup of your system, select item 5 in the installation procedure to stop the installation. Appendix A contains an example installation listing.
Stop the installation	Select item 5, `EXIT without installing any subsets`, in the installation procedure to stop the installation.
Configure *IPFM*	See Chapter 3.
Verify the installation	See information on the IVP earlier in this chapter.
Rebuild the kernel	See *Tru64 UNIX* system management documentation.
Install optional products	See the *AlphaServer Intelligent Peripheral Platform System Manager's Guide*.

De-installing Savesets

If the installation fails, and the subsets are not completely installed, use the following command to de-install the savesets:

Type

- `#setld -d IPFMALARM220 IPFMAPI220 IPFMDRIVER220 IPFMTEST220`

The following information is displayed on your screen:

- `Deleting "IPFMTEST - IPFM Fault Manager Tests" (IPFMTEST220).`
- `Deleting "IPFMAPI - IPFM Application Interface" (IPFMAPI220).`
- `Deleting "IPFMDRIVER - Alarm Panel Driver" (IPFMDRIVER220).`
- `Deleting "IPFMALARM - Event Man/Det Process" (IPFMALARM220).`

3
Configuring the IPFM Software

This chapter describes the procedures that must be performed to configure the *IPFM* software.

Configuring DECevent Software

To run the *IPFM* software with *DECevent*, *DECevent* must be set to the report type `ETM_2_RE`.

1. To check the DECevent report type, enter the following command:

   ```
   # dia shw set DEFAULT_REPORT
   ```

2. If the report type is not `ETM_2_RE`, enter the following set of commands to set the report type:

   ```
   #dia -int
   dia> sets DEFAULT_REPORT ETM_2_RE
   dia> sav sys
   dia> quit
   ```

3. Restart the *IPFM* software by running the following commands:

   ```
   # ipfm_shutdown
   # ipfm_init.d
   ```

Customizing the IPFM Configuration File

The *IPFM* configuration file contains the following information:

- Event Category - The category of event to be detected and reported.
- Event Severity - The severity level of the event.
- Event Text - The text of the event.

The *IPFM* configuration file is located in the following directory:

```
/usr/opt/IPFM/bin/online/ipfm.conf
```

The user can modify the *IPFM* configuration file in order to:

- Enable or disable the detection of events
- Change the severity of an event
- Change the text used for an event.

Note

The event severity and event text can not be modified on the "system" detected events. The severity and text will be determined from the component generating the error.

The configuration file allows the user to modify what types of events are monitored, and in some cases the severity to be used when the associated alarms are generated. For a sample `ipfm.conf` file, see Appendix 6C.

Table 3-1 shows the keywords that are defined in the `ipfm.conf` file. Comments in the file and descriptions in the table explain how the configuration file can be modified. In most cases the configuration file contains examples on the format and options of the keywords and parameters.

Table 3-1: Configuration File Keywords

Keyword	Description
Configuration Keywords	
CONTROL	Indicates whether there is an IPFM PCI control module in the system. The parameter is TRUE if there is a control module in the system and FALSE if there is not.
INDICATOR	Indicates whether there is an IPFM Alarm panel indicator module hooked up to the CONTROL module. The parameter is TRUE if there is an alarm panel attached to the system and FALSE it there is not.
ASC-LED	Defines the pattern displayed in the status LED. Compaq recommends that this parameter not be modified.
KEEP-ALIVE	Specifies the keep-alive update frequency. The parameter specifies the number of seconds between sending the alarm panel a keep-alive signal. If the keep-alive signal is reset within x seconds, the alarm panel signals a minor alarm.
SHUTOFF_SW	Detect Audible Disable switch.
AUDIBLE	Enable or Disable the audible alarm. If this parameter is set to TRUE, the alarm panel signals alarms using an audible beep. If the parameter is set to FALSE, no audible beep is generated.
POLLING	Controls the polling interval. The parameter contains the polling interval, in seconds, for the checking of File System, System Environmental, and Process Missing events. The default is 5 seconds.
TRAP-THRESHOLD	Specifies the SNMP trap threshold severity. The parameter allows the user to limit the SNMP traps that are generated. The user can specify a TRAP-THRESHOLD severity to cause IPFM to only generate SNMP traps when the alarm has an equal or greater severity than the TRAP-THRESHOLD severity.

Keyword	Description
System Events	The following four keywords monitor system events. The severity and alarm text may not be modified on these events. If one of these event types is not desired, comment out the line by inserting a # before the keyword.
CPU	Monitors CPU-related events.
DISK	Monitors DISK-related events.
NETWORK	Monitors NETWORK-related events.
MEMORY	Monitors MEMORY-related events
FS	Defines File System capacity thresholds. The user can specify a file system to be monitored by the IPFM software. The user specifies the file system mount point, the available capacity (in either bytes or as a percentage) and the associated alarm severity.
PROCESS	Monitors the system for one or more specific process names. The user can instruct IPFM to monitor the system for the presence of one or more processes. The user specifies the process name, the process's UID, the number of processes that should be running, the associated alarm severity and an optional action script to run if the number of processes is lower than expected.
System Environmental Events	The following three keywords monitor environmental events. The user can specify the associated alarm severity and alarm text in the configuration file if desired. If one of these events is not desired, comment out this line by inserting a # before the keyword.
TEMP-CPU	Monitors the temperature of the system. If the system temperature thresholds are crossed, IPFM generates a TEMP-CPU alarm.
FAN-CPU	Monitors the system fan. If the system fan is detected to fail, IPFM generates a FAN-CPU alarm.
POWER-CPU	Monitors the system power supply (valid only if there is redundant power supplies). If a failure in the redundant power supply is detected, IPFM generates a POWER-CPU alarm.
Expansion Chassis Events	The following three keywords monitor expansion chassis events. The user can specify the associated alarm severity and alarm text in the configuration file if desired. Since the expansion box is an optional component, the expansion chassis event keywords are commented out by default. To enable an expansion chassis event keyword, remove the commenting by deleting the # before the keyword
TEMP-EXT	Monitors the temperature of the external expansion box (if present). If the expansion box temperature threshold is crossed, IPFM generates a TEMP-EXT alarm.
FAN-EXT	IPFM monitors the external expansion box fan. If a fan failure is detected, IPFM generates a FAN-EXT alarm.
POWER-EXT	IPFM monitors the expansion box power supply. If a power supply failure is detected, IPFM generates a POWER-EXT alarm.

Keyword	Description
External Signals	The following three keywords represent three external signals that can be monitored. These external signals are connected to the –48Vdc power supply by a cable if the system was configured with a –48Vdc power option. The user can specify the associated alarm severity and alarm text in the configuration file if desired. Since the –48Vdc power supply is an optional component, the USER-EVENTx keywords are commented out by default. To enable a USER-EVENTx keyword, remove the commenting by deleting the # before the keyword.
USER-EVENT1	When enabled, defaults to the "Fail relay on" event description for the -48V power inverter.
USER-EVENT2	When enabled, defaults to the "Minor alarm relay on" event description for the -48V power inverter.
USER-EVENT3	When enabled, defaults to the "Major alarm relay on" event description for the -48V power inverter.

Modifying the SNMP Agent Configuration File

The SNMP Agent configuration file , `/etc/snmpd.conf`, must have the following items added to allow the Network Management System (NMS) to work with the *IPFM* target system:

1. To get started, it is suggested that the public community be set to read/write as follows:

   ```
   Community public        0.0.0.0      write
   ```

2. A community specification may be added to include the IP address of the NMS and to allow read and write privileges as follows (where `xx.xx.xx.xx` stands for the IP address of the NMS and `ipfm` is the community name):

   ```
   community ipfm          xx.xx.xx.xx  write
   ```

3. The trap community specification must be added as follows (where `xx.xx.xx.xx` stands for the IP address of the NMS):

   ```
   trap      ipfm          xx.xx.xx.xx
   ```

Configuring the IPFM SNMP Trap Style

Two SNMP trap styles are available:

- Default IPFM SNMP trap style
- Alternate IPFM SNMP trap style, for IPFM that is OSI compliant

Default IPFM SNMP Trap Style

The default SNMP trap style consists of the following variable bindings:

Table 3-2: Default SNMP Trap Variable Binding List

OID	Value	Description
1.3.6.1.4.1.36.2.15.25.5.1.1	1	Alarm ID
1.3.6.1.4.1.36.2.15.25.5.1.2	CRITICAL	"ASCII" severity
1.3.6.1.4.1.36.2.15.25.5.1.3	New critical alarm	Text description of alarm
1.3.6.1.4.1.36.2.15.25.5.1.4	ACTIVE	Alarm state (ACTIVE, ACK, or CLEAR)

alarmIndex (OID: 1.3.6.1.4.1.36.2.15.25.5.1.1)

The alarmIndex contains an alarm index. This is a unique value that can be used to further identify the alarm within the IP.

alarmSeverity (OID: 1.3.6.1.4.1.36.2.15.25.5.1.2)

The alarmSeverity is a text string containing the severity of the particular alarm: CRITICAL, MAJOR, MINOR, WARNING or INFO.

alarmDescript (OID: 1.3.6.1.4.1.36.2.15.25.5.1.3)

The alarmDescript contains the alarm text.

alarmStatus (OID: 1.3.6.1.4.1.36.2.15.25.5.1.4)

The alarmStatus is a text string containing status of the particular alarm: ACTIVE, ACK or CLEAR.

Alternate IPFM SNMP Trap Style

The Alternate SNMP trap style was created for IPFM that is OSI compliant. Using this alarm trap style simplifies the integration of IPFM with TeMIP and other Network Management Systems.

To generate the alternate style alarm, define the environment variable IPFM_NMS_ALARM to 1 prior to starting IPFM. Normally, this is done by specifying it in the ".profile " file executed at login. For example, enter the following in the /.profile file:

```
IPFM_NMS_ALARM=1
export IPFM_NMS_ALARM
```

If IPFM_NMS_ALARM is not specified, or it is set to any value other than 1, IPFM defaults to the standard alarm it sends. Note, the alarm style cannot be changed while IPFM is running. IPFM needs to be restarted to change the alarm style.

The alternate SNMP trap style has the following format:

Table 3-3: Variable Binding List

OID	Value	Description
1.3.6.1.4.1.36.2.15.25.5.1.1	1	Alarm ID
1.3.6.1.4.1.36.2.15.25.5.1.2	CRITICAL	"ASCII" severity
1.3.6.1.4.1.36.2.15.25.5.1.3	New critical alarm	Text description of alarm
1.3.6.1.4.1.36.2.15.25.5.1.4	ACTIVE	Alarm state (ACTIVE, ACK, or CLEAR)
1.3.6.1.4.1.36.2.15.25.5.1.6	1	Integer severity
1.3.6.1.4.1.36.2.15.25.5.1.7	0	Alarm probable cause
1.3.6.1.4.1.36.2.15.25.5.1.8	0	Alarm event type

The first four variables (OIDs) are the same variables generated in the default IPFM alarm. The following describes the additional three variables (OIDs).

alarmIntSeverity (OID: 1.3.6.1.4.1.36.2.15.25.5.1.6)

alarmIntSeverity is the integer representation of the severity variable. It can have the following values:

Severity	Integer Value
critical	1
major	2
minor	3
warning	4
clear	5

The value of alarmIntSeverity is based on the IPFM "ASCII" Severity and the alarm state. Alarms with a severity of "INFO" are mapped to the alarmIntSeverity of 4 (Warning). AlarmIntSeverity is set to 5 (CLEAR) when the alarm state (variable 4) is set to CLEAR. Note, a CLEAR alarm is sent with the same alarm ID as the "ACTIVE" alarm, that generated the original alarm. Since the alarm ID in both alarm messages is the same, this allows an NMS to automatically correlate alarms. The chart below displays how alarmIntSeverity is assigned.

	ACTIVE	ACK	CLEAR
critical	1	1	5
major	2	2	5
minor	3	3	5
warning	4	4	5
info	4	4	5

alarmProbableCause (OID: 1.3.6.1.4.1.36.2.15.25.5.1.7)

The alarmProbableCause indicates the cause of the alarm. Presently, this OID is not used and defaults to 0.

This field may be used in a future release of the IPFM product.

alarmEventType (OID: 1.3.6.1.4.1.36.2.15.25.5.1.8)

The alarmEventType indicates the type of alarm. Presently, this OID is not used and defaults to 0. This field may be used in a future release of the IPFM product.

Configuring the Network Management Station (NMS)

Any SNMP-compliant Network Management System can be used with IPFM. This section describes the configuration of the following:

- *ServerWORKS*
- *TeMIP*

Configuring the NMS with ServerWORKS

Install *ServerWORKS* on a Windows NT system and run IP Discovery from the Open Viewer window on the target network segment. The target nodes of interest should be visible on the displayed network topology map.

Enrolling the IPFM MIB into the ServerWORKS NMS

The following steps are required to load the *IPFM* MIB into the *ServerWORKS* NMS:

1. Copy the *IPFM* MIB (`/usr/opt/IPFM/src/event_man/ipfm_mib.my`) from the *Tru64 UNIX* system to the NMS.
2. Go into the ServerWORKS Manager main window.
3. Select the MIB Enroller option from the Tools menu pulldown.
4. Select the MIB Compiler option from the Compile menu pulldown.
5. Select the Open option from File menu pulldown.
6. Select the `ipfm.mib` file from the location where it was stored. The open menu disappears and you are returned to the MIB Compiler window.
7. Select the Enroll... option.
8. Shutdown and restart *ServerWORKS* so that the MIB will be recognized by the system.

Setting Up IPFM Alarm Traps

The following steps are required to set up the *IPFM* traps in the recommended configuration:

1. Double-click on the target node for the ServerWORKS Manager - [IP Discovery] window. This causes the System Browser window to come up for the target node.
2. Select Alarm Configuration… from the Tools menu pulldown.
3. Double-click on the “SNMP Traps” option in the “Alarm Type:” window. The “Add New SNMP Trap Alarms” window comes up.
4. Search down the “SNMP Traps:” window until you find the following IPFM traps: `ipfmcritTrap`, `ipfmmajorTrap`, `ipfmminorTrap`, `ipfmwarningTrap`, and `ipfmInfoTrap`.
5. Click on `ipfmcritTrap`. It will be displayed in the “Alarm message (can be modified):” window.
6. Select High from the “Severity” section and then click on the “OK” button.
7. Follow steps 3-6 for each of the remaining four traps (`ipfmmajorTrap`, `ipfmminorTrap`, `ipfmwarningTrap`, and `ipfmInfoTrap`), the only difference being that the severity should be set as follows:

   a. `ipfmmajorTrap` – severity Medium.
   b. `ipfmminorTrap` – severity Low.
   c. `ipfmwarningTrap` –- severity Informational
   d. `ipfmInfoTrap` – severity Informational.

8. Select each newly defined trap in the “Currently defined alarms:” window and click on the “Enable” button.

You are now ready to remotely manage your *IPFM* software on the target node through SNMP.

Enrolling the IPFM MIB into the TeMIP NMS

The following steps are required to load the *IPFM* MIB into the *TeMIP* NMS:

1. Copy the *IPFM* MIB (`/usr/opt/IPFM/src/event_man/ipfm_mib.my`) from the *Tru64 UNIX* system to the NMS.
2. Compile the MIB into *TeMIP* using the following commands:

   ```
   # mcc_tcpip_mtu ipfm_mib.my –n –s

   # mcc_msl  -Xad  -mipfm_mib.ms,1

   # mcc_ptb
   ```

3. Shutdown and restart *TeMIP* for the MIB to be recognized by the system using the following commands:

   ```
   # temip_stop

   # temip_start
   ```

4. On the AlphaServer running *IPFM*, add the following lines to the `/.profile` file. This instructs *IPFM* to send SNMP traps using the alternate SNMP trap style.

   ```
   IPFM_NMS_ALARM=1

   export IPFM_NMS_ALARM
   ```

5. Restart IPFM using the following commands:

   ```
   # ipfm_shutdown

   # ipfm_init.d
   ```

4
Understanding IPFM

The following diagram (Figure 4-1) contains the *IPFM* components, which are described within this chapter. The Database Manager, Event Manager, and menu are separate application level processes. The Event Manager includes the SNMP Subagent, Alarm Panel Manager, and Database Manager sub-components. The `ipap` device driver executes at the kernel level.

Figure 4-1: IPFM Components

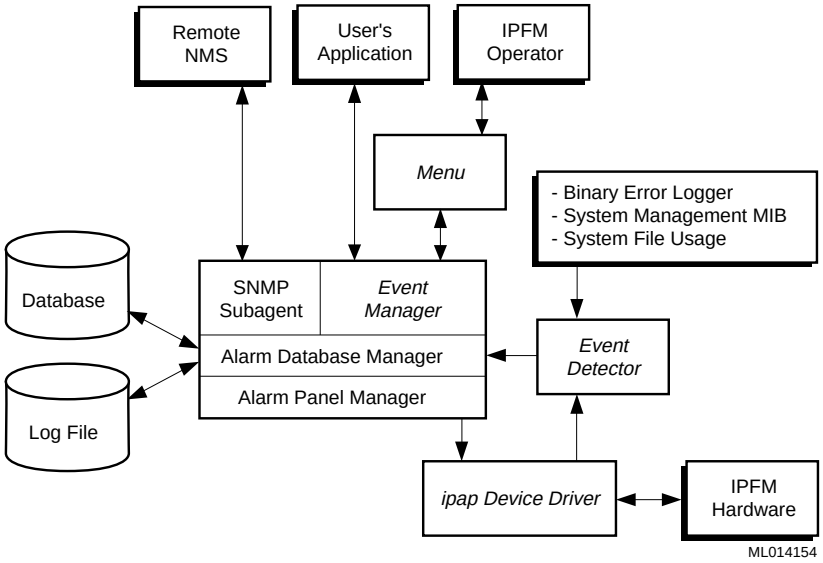


Figure 4-1 shows the Event Detector passing system and *IPFM* fault events to the Alarm Database Manager. The Alarm Database Manager updates its database and requests the Alarm Panel Manager to set an alarm indicator. The database is visible to the remote Network Management Station (NMS), user application program interface, and *IPFM* operator interface, which can be used to set, acknowledge, and clear alarms.

IPFM Event Manager

The Event Manager is the core of the *IPFM* system. It has three components: the Alarm Database Manager, the SNMP Subagent, and the Alarm Panel Manager, which are described in the following sections.

Alarm Database Manager

The Alarm Database Manager manages the *IPFM* alarm database and private MIB data. It interfaces with the user menu/display, Event Detector, and Host User Application through shared memory and with the SNMP Subagent and Alarm Panel Manager through direct routine calls.

Interface to the Event Detector

The Event Detector sends significant events to the Alarm Database Manager. These events include the generic platform system events and the external events identified in the `ipfm.conf` file.

Interface to the User Menu/Display

The user menu/display consists of the menu and display functions. The menu function allows the operator interface to set, clear, and acknowledge events of any supported severity and may also request the status of the alarm indicator panel. The display function allows the display of all alarm events contained in the alarm database.

Application Program Interface (API)

The API is the interface that allows user applications to set, acknowledge, and clear uniquely defined events. The API sends these events to the Alarm Database Manager through a shared memory interface.

SNMP Subagent

The SNMP Subagent interfaces to the *Tru64 UNIX* SNMP Master Agent. The Subagent allows any network management station that has SNMP management capability to manage alarmed event data in the *IPFM* private management information base (MIB) extension.

Table 4-1 shows the *IPFM* private Management Information Base (MIB) conceptual data structure.

Table 4-1: MIB Conceptual Data Structure

Variable Name	Data Type	Description
IpfmProductID	OBJECT IDENTIFIER	The value of this object is the registered object identifier for this product. A value of (0,0) should be returned if unsupported
NodeDescr	ASCII STRING	A textual description of the particular node within the IP system.
NodeNumAlarms	INTEGER	The current number of alarms on this particular IP node.
NodeStatus	ASCII STRING	The alarm status of this node (Critical, Major, Minor, Warning, Info or Normal).
AlarmIndex 1	INTEGER	A unique value for each alarm. Values of this object range from 1 to nodeNumAlarms *n*, where *n* is the corresponding value of nodeIndex.
AlarmSeverity 1	ASCII STRING	The severity of the particular alarm: CRITICAL, MAJOR MINOR, WARNING, or INFO.
AlarmDescript 1	ASCII STRING	The text description describing the particular alarm.
alarmStatus 1	INTEGER	The status of this alarm entry. Possible values are clear, acknowledge, or active.
alarmTimeStamp 1	ASCII STRING	A character string indicating the time of the alarm.
alarmIntSeverity 1	INTEGER	The Severity in an integer format
alarmProbableCause 1	INTEGER	The cause of the alarm. This variable is currently not used.
alarmEventType 1	INTEGER	The alarm type. This variable is currently not used
. . .	. . .	. . .
alarmIndex n	INTEGER	A unique value for each alarm. Values of this object range from 1 to nodeNumAlarms *n*, where *n* is the corresponding value of nodeIndex.
alarmSeverity n	ASCII STRING	The severity of the particular alarm: CRITICAL, MAJOR MINOR, WARNING or INFO.
alarmDescript n	ASCII STRING	The text description describing the particular alarm.
alarmStatus n	INTEGER	The status of this alarm entry. Possible values are clear, acknowledge, or active.
alarmTimeStamp n	ASCII STRING	A character string indicating the time of the alarm.
alarmIntSeverity n		The Severity in an integer format
alarmProbableCause n	INTEGER	The cause of the alarm. This variable is currently not used.
alarmEventType n	INTEGER	The alarm type. This variable is currently not used
NodeClrAllAlarms	INTEGER	Variable used to clear all outstanding alarms from the NMS
NodeAckAllAlarms	INTEGER	Variable used to acknowledge all outstanding alarms from the NMS
NodeRepostAllAlarms	INTEGER	Variable used to cause all outstanding alarms to re-post their associated SNMP traps

SNMP Traps

There are five different types of enterprise specific traps generated by the *IPFM* software. They are as follows:

ipfmCritTrap - This trap is the *IPFM* critical alarm trap. It is generated when a critical alarm condition occurs, and it contains the severity, description, and status of the critical alarm condition.

ipfmMajorTrap - This trap is the *IPFM* major alarm trap. It is generated when a major alarm condition occurs, and it contains the severity, description, and status of the major alarm condition.

ipfmMinorTrap - This trap is the *IPFM* minor alarm trap. It is generated when a minor alarm condition occurs, and it contains the severity, description, and status of the minor alarm condition.

ipfmWarningTrap - This trap is the *IPFM* warning alarm trap. It is generated when a warning alarm condition occurs, and it contains the severity, description, and status of the warning alarm condition.

ipfmInfoTrap - This trap is the *IPFM* info trap. It is generated when an information alarm occurs, and it contains the severity, description, and status of the information condition.

These traps can be monitored at different severity levels through the setup procedures of most Network Management Stations. An example setup for a Network Management Station (NMS) running *ServerWORKS* is described in Chapter 3.

IPFM SNMP Trap Examples

Each *IPFM* SNMP trap has three varbinds (attached variable bindings) associated with it. The first variable is the `alarmSeverity`, the second is the `alarmDescript`, and the third is the `alarmStatus`. These entities are described in Table 4-1.

When an *IPFM* SNMP trap is received (as above), the node object in the ServerWORKS Manager - [IP Discovery] window will have a red alarm bell appear as attached to the node. This indicates that the node has received a trap that can be seen in detail through the “View Alarm” option of the “Actions” pulldown menu from the ServerWORKS Manager - [IP Discovery] window. The *IPFM* SNMP-specified traps will have a small color box associated with the alarm in the “Alarm Viewer” window. The colors correspond with the alarm severity set earlier and are as follows:

1. Critical alarm - red.
2. Major alarm - yellow.
3. Minor alarm - green.
4. Informational - blue.

Examples of critical and major alarm conditions as seen expanded in the “Alarm Viewer” window follow:

```
 05/05/97 15:52:47 asip2b.zko.dec.com SNMP TRAP : enterpriseSpecific from
asip2b.zko.dec.com (16.126.64.64)

        05/05/97 15:52:47

        SNMP TRAP : enterpriseSpecific from asip2b.zko.dec.com (16.126.64.64)

        sysUpTime  :  3days:2hours:3mins:31secs

        Enterprise :  [1.3.6.1.4.1.36.2.15.25]   ipfm

        specific   :  [199]

        varbind 1    :  1.3.6.1.4.1.36.2.15.25.5.1.1 = 5

        varbind 2    :  1.3.6.1.4.1.36.2.15.25.5.1.2 = CRITICAL

        varbind 3    :  1.3.6.1.4.1.36.2.15.25.5.1.3 = test

        varbind 4    :  1.3.6.1.4.1.36.2.15.25.5.1.4 = ACTIVE

05/05/97 15:52:53 asip2b.zko.dec.com SNMP TRAP : enterpriseSpecific from
asip2b.zko.dec.com (16.126.64.64)

        05/05/97 15:52:53

        SNMP TRAP : enterpriseSpecific from asip2b.zko.dec.com (16.126.64.64)

        sysUpTime  :  3days:2hours:3mins:37secs

        Enterprise :  [1.3.6.1.4.1.36.2.15.25]   ipfm

        specific   :  [200]

        varbind 1    :  1.3.6.1.4.1.36.2.15.25.5.1.1 = 6

        varbind 2    :  1.3.6.1.4.1.36.2.15.25.5.1.2 = MAJOR

        varbind 3    :  1.3.6.1.4.1.36.2.15.25.5.1.3 = check

        varbind 4    :  1.3.6.1.4.1.36.2.15.25.5.1.4 = ACTIVE
```

Alarm Panel Manager

The Alarm Panel Manager tests the basic integrity of the alarm subsystem when its initialization routine is called. The tests performed do not require operator intervention. The number of tests may be restricted by parameters in the *IPFM* configuration file. For example, if these parameters define that there is no attached indicator panel, then the alarm indicator panel cable will not be tested.

IPFM Event Detector

The Event Detector provides the detection and notification of significant events to be displayed on the alarm indicator panel. The Event Detector determines the classes of events to monitor and report by parsing the *IPFM* configuration file, which contains a section detailing the event categories to be reported and the severity of the alarms. Any detected events in these categories will cause an alarm to be generated.
Table 4-2 provides a list of the monitored event categories and their default severity.

Table 4-2: Event Categories

Category	Event	Severity
System	CPU	Critical or Major
	Disk	Critical or Major
	Network	Critical or Major
	Memory	Critical or Major
Environmental	Temperature (processor)	Critical *
	Fan failure (processor)	Major *
	Power supply failure	Major *
	Expansion chassis temperature	Critical *
	Expansion chassis fan failure	Major *
	Expansion chassis power supply failure	Critical *
	Cabinet inverter failure (or user definable)	Critical *
	Cabinet inverter AC failure (or user definable)	Critical *
	Cabinet inverter DC failure (or user definable)	Critical *
File system	File system nearly full	Minor *
	Indicator module failure	Minor *
Process monitoring	Process missing	User definable
* User definable		

The following Event Categories from Table 4-2 are discussed in greater detail:

- System events
- File system events
- Process monitoring events

System Events

The “system” type events (CPU, disk, memory and network) are obtained by using the *DECevent* product to translate the binary error log events that are written into the `binary.errlog` error log file.

Table 4-3 shows the correlation between the *IPFM* Event Category (CPU, Disk, Network and Memory) and the binary event logger “event type” category. This illustrates which binary event types are associated with the IPFM event categories.

Table 4-3: Binary Event Types

Category	Binary Event Logger Type
CPU	CPU machine checks
	Device
	Adapter
	Bus
	ASCII
	Stray Interrupt
	Console
	Generalized Machine State
	Panic
DISK	Disk
	Tape
	SCSI
	CAM
	Fibre Channel
	SWXCR RAID Controller
	CI PPD
	LSM
	Advanced File System
NETWORK	Network
MEMORY	Soft/Hard Memory

File System Events

For the File system nearly full event, the user can specify a file system (specified by the mount point), and a threshold (specified in either bytes or percentage) to cause File System nearly full alarms. If a specified threshold is crossed, the associated event is generated and passed to the Alarm Database Manager. If the threshold is crossed again (below the threshold), a CLEAR event is sent to the Event Manager.

It is possible to specify multiple levels of alarms in the configuration file for the file system nearly full alarms. This allows the user to specify different levels of alerting file system usage problems.

If several thresholds are set for the same file system, when the first threshold is crossed (MINOR alarm for example), a SET event is sent to the event manager. When the second threshold is crossed (MAJOR alarm for example), a CLEAR event is generated to clear the MINOR alarm, and a SET event is generated to create the MAJOR alarm. This prevents a single file system from having two alarms with different severity. The following example shows how File system thresholds are defined in the ipfm.conf configuration file:

```
# Monitor the root partition for usage less than 10%
# FS     /      10%    MAJOR
#
# Monitor the /usr partition for usage less than 20,000 bytes
# FS     /usr   20000  CRITICAL
#
FS      /        5%     MINOR
FS      /        2%     MAJOR
FS      /usr     10%    MINOR
FS      /usr     2%     MAJOR
```

Process Monitoring Events

IPFM V2.1a and above incorporates the ability to optionally detect missing processes. The processes to monitor are specified in the configuration file, `ipfm.conf`. Every five seconds (default), the "polled_event_detector" checks whether the processes specified in the configuration file are still running. If any of these processes are not running, an alarm of the specified type is generated. This alarm is only generated once. If the alarm-type is NONE, the alarm is displayed on the IPFM menu, but no SNMP trap is generated. If a script is specified in the configuration file, it is also run upon detection of a missing process. This allows the user to automatically restart the missing process. The missing process can be restarted manually as well.

To implement this functionality, a variable has been added to the configuration file to specify the processes to monitor, as follows:

```
PROCESS  process-name  uid  count  alarm-type  script-to-run
```

Process_name	Name of the process to monitor. Use the full path name without arguments. For example, `/usr/sbin/snmpd`
uid	uid of the process.
count	Number of processes running with process_name.
alarm-type	Alarm to generate if the process disappears: INFO, WARNING, MINOR, MAJOR, CRITICAL, NONE
script-to-run	Specifies the user-written script to restart the process. (OPTIONAL parameter)

IPFM Alarm Subsystem

The *IPFM* alarm subsystem consists of the alarm indicator panel, the alarm control module, the ISA bus expansion chassis, and the *IPFM* alarm subsystem device driver. The *IPFM* alarm subsystem provides two functions: fault detection and an alarm display. The fault detection is in addition to that provided through normal system resources. Sensors within the *IPFM* alarm subsystem provide fault detection for airflow, chassis temperature, and chassis voltage. User defined sensors can be provided as well. To be used, all sensors must be enabled by making entries in the file `ipfm.conf`. Because hardware connections for user defined sensors are not present in the default *IPFM* system configuration, these connections must be specified when the *IPFM* platform is ordered or connected at the customer's site.

The *IPFM* alarm subsystem provides three severity levels of visual and audible alarm notification: critical, major, and minor. Severity levels are described in "IPFM Event Detector" earlier in this chapter. There are two levels of severity that do not result in visual or audible alarms: warning and informational.

The *IPFM* alarm subsystem can be tested (offline) using the `aptest` utility described in Appendix E.

IPFM Hardware

IPFM hardware consists of an alarm control module, an alarm indicator panel, a connecting cable, an alarm sensor module, and sensor lines. The alarm control module plugs into a PCI bus slot. It contains:

- connectors for inputs from the ISA expansion chassis and - 48v inverter (or 3 user-defined inputs)
- connector for attaching to the alarm indicator panel
- logic to manage the setting of alarms and reading of fault sensor lines

The alarm indicator panel, mounted in the system cabinet, contains these components:

- a light-emitting diode (LED) used as a module okay indication
- three LEDs for visual alarm notification
- an audible alarm indicator
- a keep-alive timer
- an ASCII status display
- battery backup
- dry contacts

Fault detection and alarm displays are normally separate functions within the *IPFM* alarm subsystem. For example, a fan fault may result in a visual and audible alarm but only after *IPFM* applications have processed the event and decided that there is not already a current alarm notification with the same severity.

The detection and alarm notification for failing *IPFM* components is an exception to this mode of operation. The *IPFM* Subsystem hardware contains a timer. After enabling this timer, the *IPFM* Database Manager must periodically restore the timer's countdown value so that it will not expire. When the timer expires, the hardware itself initiates alarm notification. If the *IPFM* components are no longer functioning due to a power outage, battery backup provides the power for alarm notification.

Note

Batteries on the alarm indicator panel are chargeable. These batteries will drain down when the system has been powered off for some time, such as during shipment. While they are recharging a "b" is displayed in the status display. This display overrides any attempt by *IPFM* system software to change its state until the batteries become charged.

Optionally, the hardware can be removed from the *IPFM* product by editing the configuration file `ipfm.conf` used by the `event_man` and `event_det` applications. The variable following the parameters CONTROL and INDICATOR should be changed from TRUE to FALSE so that the modules are not expected to be present. Refer to the *AlphaServer Intelligent Peripheral Platform Hardware Guide* for more information.

Alarm Indicator Panel

The alarm indicator panel contains the following:

- Visual indicators
- Audible indicator
- Battery backup logic
- Keep-alive function
- Dry contacts

Visual Indicators

The alarm indicator panel has three alarm LED indicators. These LED indicators are defined as follows:

- Critical (red LED)
- Major (red LED)
- Minor (amber LED)

The LED for a particular level of severity will be lit as long as at least one event of that level is active. Multiple LEDs may be simultaneously lit.

The alarm indicator panel has two other indicators used for status information. These are the following:

- Status display
- Alarm indicator panel OK LED (green)

The status display is used to output status as a result of the diagnostic testing and to display other status information as shown in the following tables.

The OK LED is lit after a successful reset and will remain lit until either an unsuccessful reset or a loss of battery power occurs.

Note

The alarm indicator panel should not be permanently removed without de-configuring it in the *Intelligent Peripheral Fault Manager (IPFM)* config file (`ipfm.conf`). Refer to Chapter 3.

Table 4-4 describes the normal conditions of the alarm indicator panel when the *AlphaServer* 1000A is powered on and the *IPFM* software is running.

Table 4-4: Alarm Panel Under Normal Conditions

Condition	OK LED	Minor Alarm LED	Status Display	Audible Alarm
Fully discharged batteries	On	Off	“b”	Off
Fully charged batteries	On	Off	“Rotating bar”	Off
Partially charged batteries	On	Off	“b”	Off
Software Diagnostics	On	Off	“d”	Off
Software Test	On	Off	“t”	Off

If the *AlphaServer* 1000A system looses power while the *IPFM* software is running, Table 4-5 describes the possible conditions of the alarm indicator panel.

Table 4-5: Alarm Panel During System Power Loss.

Condition	OK LED	Minor Alarm LED	Status Display	Audible Alarm
Fully discharged batteries	Off	Off	blank	Off
Fully charged batteries	On	On	“p”	On
Partially charged batteries	On	On	“b”	On

Audible Indicator

The audible indicator has three distinct sounds associated with the three levels of alarming. This indicator is controlled by software through the alarm control module, or by a disable switch on the alarm indicator panel itself. The audible indicator will sound at the level of the most severe alarmed event that is currently active. No more than one level of audible alarm can be enabled at any one time. The disable switch turns off the current audible alarm until a new event occurs that is of equal or greater severity level than the highest level alarm currently active, or until the event causing the audible alarm is cleared and another alarmed event is pending.

The three audible alarm levels are defined as follows:

- Critical is two beeps, the first separated from the second by 0.5 seconds or less. This double beep pattern is repeated every 1.5 seconds.
- Major is one beep every 1.5 seconds.
- Minor is one beep every 5 seconds.

The audible beep can be disabled by setting the AUDIBLE parameter to FALSE in the `ipfm.conf` configuration file.

Battery Backup Logic

The battery backup logic is designed to keep the current indicator status functioning for the life of the batteries (approximately 60 to 120 minutes) in the event of a power failure. It provides an indication if the batteries are low by displaying a “b” in the status display.

Note

Compaq recommends that battery replacement be performed every two to two and a half years as preventative maintenance.

Keep-Alive Function

The alarm indicator panel houses a timer that is used to ensure that the CPU controlling it is still running. The function is enabled by software, which starts a timer. As long as software resets the timer before expiration, a “rotating bar” is displayed in the status display, otherwise a minor alarm is generated. A minor alarm will also be generated if power to the alarm indicator panel is lost and a “p” will be displayed in the status display.

IPAP Device Driver

The ipap device driver is automatically installed with the rest of the *IPFM* product using the UNIX system setld utility. The ipap device driver is a *Tru64 UNIX* PCI bus driver with the following characteristics:

- it is dynamically configured (i.e. loaded and removed)
- it allows only one open of the driver per application type (event manager, event detector and aptest)
- it allows the aptest utility to access up to eight module sets including eight alarm control modules and eight indicator modules (note, online applications are limited to one module set).

Dynamically Configured

The ipap device driver is started without rebuilding and rebooting the operating system. The driver can also be removed from the operating system without shutting the system down. Normally the steps performed will occur transparently to the user with the components of the *IPFM* product starting or shutting down in sequence.

Before the *IPFM* product is started, the status display of its alarm indicator panel will normally be blank. After the *IPFM* product is started, the status display will normally display a changing pattern. Between the blank and changing pattern, the ipap device driver sets the status display to "d". If the ipap device driver is started independently of the *IPFM* application, the "d" should become visible to the user. If it does not, read the /var/adm/messages file or the /var/adm/syslog.dated/*date*/user-log file for an indication of the problem. When the ipap device driver is removed, the status display again becomes blank.

One Open per Application

The *IPFM* online product applications support only one hardware module set including one alarm control module and one alarm indicator panel. However the /dev directory will contain three or more entries for the ipap device driver. These /dev entries do not correspond to multiple controllers or devices but are used by the ipap driver to keep track of application connections as follows:

- /dev/ipap0 - opened by event_det application
- /dev/ipap1 - opened by event_man application
- /dev/ipap2 - opened by off-line aptest diagnostic test

The aptest utility can then be used to test more than one set of modules, as described in Appendix E. The online environment of the ipap device driver can be restored after using the diagnostic as follows:

Command	Effect
cd /usr/sys/io/IPAP100	Go to the ipap device driver directory
/sbin/sysconfig -u ipap	Stop ipap execution
/sbin/sysconfigdb -d ipap	Remove ipap entries from /etc/sysconfigtab)
mv sysconfigtab_online sysconfigtab	Restore one module set to directory
/sbin/sysconfigdb -a -f sysconfigtab ipap	Restore one module set to /etc/sysconfigtab)
/usr/opt/IPFM/bin/online/ipfm_init.d	Put the *IPFM* online

IPFM Event Logs

The *IPFM* event log files can be found in the `/usr/opt/IPFM/log` directory. The log file stores all activity related to any event, tracking the Sets, Acknowledgements, and Clears.

The log file is created new each day at midnight The name of the log file has the following format:

```
IPFM_event_log.dd_mmm_yyyy
```

For example:

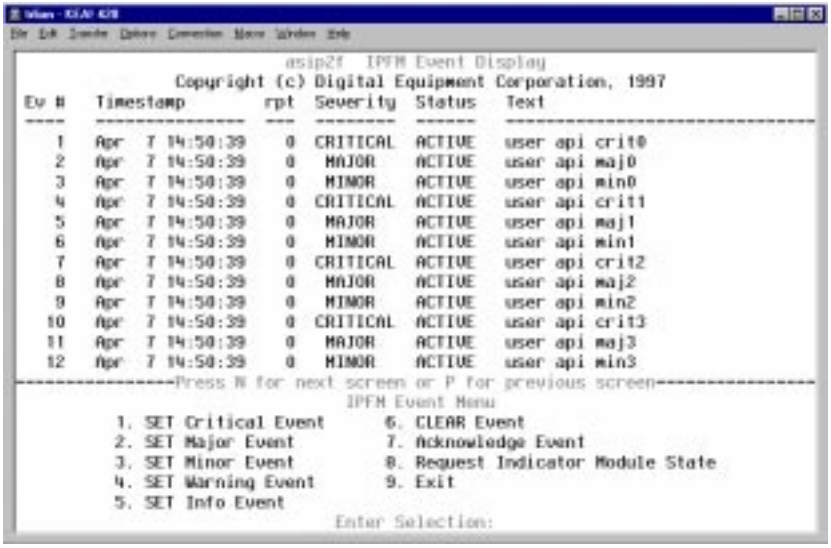
```
IPFM-event_log.01_Apr_1999
```

5
Alarm Utility Operator Interface

Overview

The *IPFM* operator interface is a management utility that allows the user to view all alarms that are present on a system. The interface is written so that any character cell terminal can be used. Figure 5-1 shows the interface display.

Figure 5-1: Operator Interface Screen Display



The top portion of the screen shows the current alarms, and the bottom of the screen contains an operations menu, labeled “IPFM Event Menu.” The following columns are included in the top portion of the display:

1. Event Number - A numeric representation of the event. This number will be used to Acknowledge or Clear an event.
2. Timestamp - The time that the event originally occurred.
3. Repeat Count - The number of times an exact event re-occurred since it was first detected.
4. Severity - The event severity. The possible levels of severity are CRITICAL, MAJOR MINOR, WARNING and INFO.
5. Status - The event status. The possible status conditions are ACTIVE and ACK (acknowledged).

6. Alarm Text - The text of the alarm description.

Accessing the Operator Menu/Event Display

To access the Operator Menu/Event Display, you must have superuser privileges. Invoke the menu using the following command:

```
# /usr/opt/IPFM/bin/online/menu
```

Using the Operator Menu/Event Display

The display will re-size itself depending on the number of rows and columns that you are using for your terminal. The top portion of the display, showing current alarm events within the system, changes size depending on the size of the terminal. The bottom portion of the display, showing the operations menu, is a fixed size. The available menu functions are as follows:

2. Set Critical Event
3. Set Major Event
4. Set Minor Event
5. Set Warning Event
6. Set Info Event
7. Clear Event
8. Acknowledge Event
9. Request Indicator Module State
10. Exit

If the event text string is longer than the available size, the text string is truncated, and a `$` will be placed in the last column. If more than one screen of events exist, the separator will show the following line:

```
Press N for next screen or P for previous screen
```

The user can press `N` <RETURN> to view the next screen of events, or `P` <RETURN> to view the previous screen of events.

Set Event

To create an event on the system, IPFM Event Menu options 1, 2, 3, 4 and 5 can be used to generate a Critical, Major, Minor, Warning or Info event respectively. After selecting menu option 1, 2, 3, 4, or 5, the user will be presented with the following prompt:

`Enter text for` ***`severity`*** `Alarm:`

where ***`severity`*** is the severity level of the event, for example `critical`.

The user then enters the alarm text and presses the Return key, which generates an alarm of the selected severity with the specified text within the system. A log entry showing the new event will be placed in the *IPFM* alarm log file.

Clear Event

To clear an event that is listed in the display portion of the screen, select item 4 from the menu. The user is presented with the following prompt:

```
Enter Event Number:
```

The user enters the event number (listed in the left hand column of the display) that is associated with the event to be cleared. The selected event is cleared from the system, and a log entry showing the cleared event will be placed in the *IPFM* alarm log file. The user can also enter * to clear all events.

Acknowledge Event

To acknowledge an event that is listed in the display portion of the screen, select item 5 from the menu. The user is presented with the following prompt:

```
Enter Event Number:
```

The user enters the event number (listed in the left hand column of the display) that is associated with the event to be acknowledged. The selected event is acknowledged, and the status field of the display will change from ACTIVE to ACK. A log entry showing the cleared event will be placed in the *IPFM* alarm log file. The user can also enter * to acknowledge all events.

NOTE:

Only ACTIVE alarms are reflected on the *IPFM* alarm indicator panel (both audible and visual).

Request Alarm Indicator Panel Status

To request the status of the alarm indicator panel, select item 8 from the menu. The screen displays a line similar to the following:

```
Indicator Module LEDs lit are:
```

The list that follows that line will include all the alarm indicator panel LEDs that are lit. Any combination, or none, of the three LEDs may be displayed.

Exit from Menu/Event Display

To Exit the Operator Menu/Event Display, select item 9 from the menu.

6
User Application Program Interface

Overview

The *IPFM* user application program interface allows user applications to set, clear, or acknowledge alarms which turn on or off the individual LEDs on the alarm indicator panel and the audible alarm.

The `ipfm_user_api` is a sharable library that resides in the `/usr/shlib` directory. The alarm condition functions can be linked into any user-written application.

API Commands

Valid alarm commands are defined in the `/usr/opt/IPFM/src/include/ipfm_alarm.h` file.

IPFM User API Routine Definition

Name

`ipfm_user_api` - IP Fault Manager communication routine

Library

IPFM C Library (`libipfm.so`)

Synopsis

```
#include <ipfm_alarm.h>

int ipfm_user_api(
        int command,
        const char *str);
```

Parameters

`command` Specifies the *IPFM* command to execute

SET_CRITICAL_ALARM	Creates a CRITICAL alarm. The alarm will be set, and an event is forwarded to any configured SNMP management station.

ACK_CRITICAL_ALARM	Acknowledges a CRITICAL alarm. The status of the alarm will change from ACTIVE to ACK, and the acknowledge is forwarded to any configured management station.
CLEAR_CRITICAL_ALARM	Clears a CRITICAL alarm. The alarm will be cleared, and the clear event is forwarded to any configured SNMP management station.
SET_MAJOR_ALARM -	Creates a MAJOR alarm. The alarm will be set, and an event is forwarded to any configured SNMP management station.
ACK_MAJOR_ALARM	Acknowledges a MAJOR alarm. The status of the alarm will change from ACTIVE to ACK, and the acknowledge is forwarded to any configured SNMP management station.
CLEAR_MAJOR_ALARM	Clears a MAJOR alarm. The alarm will be cleared, and the clear event is forwarded to any configured SNMP management station.
SET_MINOR_ALARM	Creates a MINOR alarm. The alarm will be set, and an event is forwarded to any configured SNMP management station.
ACK_MINOR_ALARM	Acknowledges a MINOR alarm. The status of the alarm will change from ACTIVE to ACK, and the acknowledge is forwarded to any configured SNMP management station.
CLEAR_MINOR_ALARM	Clears a MINOR alarm. The alarm will be cleared, and the clear event forwarded to any configured SNMP management station.
SET_WARNING_ALARM	Creates a WARNING alarm. The alarm will be set, and an event is forwarded to any configured SNMP management station.
ACK_WARNING_ALARM	Acknowledges a WARNING alarm. The status of the alarm will change from ACTIVE to ACK, and the acknowledge is forwarded to any configured SNMP management station.
CLEAR_WARNING_ALARM	Clears a WARNING alarm. The alarm will be cleared, and the clear event is forwarded to any configured SNMP management station.
SET_INFO_ALARM	Creates an INFO informational alarm. The alarm will be set, and an event is forwarded to any configured SNMP management station.
ACK_INFO_ALARM	Acknowledges an INFO informational alarm. The status of the alarm will change from ACTIVE to ACK, and the acknowledge is forwarded to any configured SNMP management station.
CLEAR_INFO_ALARM	Clears an INFO informational alarm. The alarm will be cleared, and the clear event is forwarded to any configured SNMP management station.

CLR_ALL_ALARMS	Clear all outstanding alarms. All outstanding alarms will be cleared from the IPFM alarm database. An SNMP trap will be forwarded to any configured network management station (NMS) for each alarm that is cleared.
ACK_ALL_ALARMS	Clear all outstanding alarms. All outstanding alarms will be cleared from the IPFM alarm database. An SNMP trap will be forwarded to any configured network management station (NMS) for each alarm that is cleared.
GET_NUM_ALARMS	Return the number of outstanding alarms present in the IPFM alarm database.
GET_ALL_ALARMS	Return the alarm text (as displayed in the alarm log file, for all outstanding alarms
REQ_STATE	Returns the current state of the alarm indicator LEDs. The `str` argument will receive a string that explains which, if any, LEDs are lit on the alarm indicator panel. The result will be formatted as follows (in the case of a CRITICAL and a MAJOR alarm): "STATE=CRITICAL,MAJOR"
REQ_VERSION	Returns the current version of the *IPFM* layered product software. The result will be formatted as follows: "VERSION=V2.0"
`str`	In the case of a SET, ACK, or CLEAR command, the `str` field contains the alarm text. The maximum allowable size of this field can be 89 characters; any additional characters will be truncated. Legal characters include letters, numbers, and spaces. Punctuation is not recognized.
`str`	In the case of a REQ_* command, the `str` field will receive the output text from the command. The maximum character output is limited to 30 characters.

Description

The ipfm_user_api() function communicates with the IP Fault Manager layered product. Fault Events can be created (SET), removed (CLEAR), or acknowledged (ACK).

Return Values

The ipfm_user_api() function returns one of the following values:

[EACCES] The calling process does not have the appropriate privilege.

[IPFM_SUCCESS] Successful completion.

[IPFM_LOCK_RETRY_EXCEEDED] Cannot acquire shared memory lock retries exceeded.

[IPFM_INV_ALARM_MSG] Invalid API parameter.

[IPFM_INV_MSG_BUF] API buffer too small.

[IPFM_NO_TEXT_MATCH] No matching alarm text.

[IPFM_NO_ADM] The Alarm Database Manager (ADM) is not running.

[IPFM_NO_ADM_RESPONSE] The ADM did not respond in the allotted time.

[IPFM_NOT_UNIQUE] The specified event text was not unique.

[IPFM_OBSOLETE] An obsolete command was issued.

[IPFM_ERROR] Internal *IPFM* error - See `syslog:user.log` for more information.

[IPFM_INV_PARAM] Invalid parameter.

Example: Setting and Clearing Critical Alarms

In the following examples, the alarm status is first set to `CRITICAL` and then set to `CLEAR`.

SET_CRITIAL_ALARM

```
#include <stdio.h>
#include "ipfm_alarm.h"

main()
{
    int stat = 0;
    char buffer_address[80] = ">> IPFM now at V. 2.0";

    stat = ipfm_user_api(SET_CRITICAL_ALARM, buffer_address);
    if (stat != 0)
        printf("unsuccessful - error = %d\n", stat);
    else
        printf("successful\n");
     }
```

CLR_CRITICAL_ALARM

```
#include "ipfm_alarm.h"
#include <stdio.h>
main()
{
    int stat = 0;
    char buffer_address[80] = ">> IPFM now at V. 2.0";

    stat = ipfm_user_api(CLR_CRITICAL_ALARM, buffer_address);
    if (stat != 0)
        printf("unsuccessful - error = %d\n", stat);
    else
        printf("successful\n");
     }
```

IPFM_GET_EVENT Routine Definition

Name

`ipfm_get_event` – Get notification of an alarm when it is generated.

Library

IPFM C Library (`libipfm.so`)

Synopsis

```
#include <ipfm_alarm.h>

int ipfm_get_event(
        char *event_text,
        int length);
```

Parameters

`event_text` Specifies a character string buffer that will receive the IPFM alarm string.

Length – Specifies that size of the buffer supplied in the event_text argument.

Description

The ipfm_get_event() function is a synchronous routine that will wait until an IPFM alarm (SET, ACK or CLEAR) is generated, and return the event text string in the event_text buffer. The format of the event will be identical to the event entry written to the IPFM event log file. If the event text is longer than the length specified in the "length" argument, the event text is truncated to "length" characters. Otherwise a NULL terminated event text string is returned in the event_text character string. If an event arrives between the time the ipfm_get_event routine returns and the next time ipfm_get_event is called, the next event in chronological order is returned immediately.

An example of the alarm text is shown here:

```
SET     Feb 16 16:04:02    CRITICAL    Test alarm
```

Return Values

The ipfm_get_event() function returns one of the following values:

[EACCES] The calling process does not have the appropriate privilege.

[IPFM_SUCCESS] Successful completion.

[IPFM_LOCK_RETRY_EXCEEDED] Cannot acquire shared memory lock retries exceeded.

[IPFM_NO_ADM] The ADM is not running.

[IPFM_NO_ADM_RESPONSE] The ADM did not respond in the allotted time.

[IPFM_ERROR] Internal IPFM error - See `syslog:user.log` for more information.

[IPFM_INV_PARAM] Invalid parameter.

A
Sample Installation Script

```
# .setld -l kit

*** Enter subset selections ***

The following subsets are mandatory and will be installed automatically
unless you choose to exit without installing any subsets:

      * IPFMALARM - Event Man/Det Process
      * IPFMAPI - IPFM Application Interface
      * IPFMDRIVER - Alarm Panel Driver

The subsets listed below are optional:

     There may be more optional subsets than can be presented on a single
     screen. If this is the case, you can choose subsets screen by screen
     or all at once on the last screen. All of the choices you make will
     be collected for your confirmation before any subsets are installed.

     1) IPFMTEST - IPFM Fault Manager Tests

Or you may choose one of the following options:

     2) ALL mandatory and all optional subsets
     3) MANDATORY subsets only
     4) CANCEL selections and redisplay menus
     5) EXIT without installing any subsets

Enter your choices or press RETURN to redisplay menus.

Choices (for example, 1 2 4-6): 2

You are installing the following mandatory subsets:

        IPFMALARM - Event Man/Det Process
        IPFMAPI - IPFM Application Interface
        IPFMDRIVER - Alarm Panel Driver

You are installing the following optional subsets:

IPFMTEST - IPFM Fault Manager Tests
```

```
Is this correct? (y/n): y

Checking file system space required to install selected subsets:

File system space checked OK.

4 subset(s) will be installed.

Loading 1 of 4 subset(s)....

IPFMALARM - Event Man/Det Process
   Copying from kit (disk)
   Verifying

Loading 2 of 4 subset(s)....

IPFMDRIVER - Alarm Panel Driver
   Copying from kit (disk)
   Verifying

Loading 3 of 4 subset(s)....

IPFMAPI - IPFM Application Interface
   Copying from kit (disk)
   Verifying

Loading 4 of 4 subset(s)....

IPFMTEST - IPFM Fault Manager Tests
   Copying from kit (disk)
   Verifying

4 of 4 subset(s) installed successfully.


Configuring "IPFMALARM - Event Man/Det Process" (IPFMALARM220)

Configuring "IPFMDRIVER - Alarm Panel Driver" (IPFMDRIVER220)

 Is there an Alarm Sub-system Control Module on the PCI Bus ([y]/n): y

 Are the inverter fault lines attached to this Control Module? (y/[n]) n


 Is there an EXPANSION BOX attached to this processor? (y/[n]) y

 Changing ipfm.conf to include it.


 Is there an Alarm Sub-system Indicator Module ([y]/n): y

 Do you want setld to start the IPFM product at this time? (y/[n]) y
```

```
 setld is executing ipfm_init.d now.

start the event manager

Configuring "IPFMAPI - IPFM Application Interface" (IPFMAPI220)

Configuring "IPFMTEST - IPFM Fault Manager Tests" (IPFMTEST220)
# exit
```

B
Files Installed on the System

Table B-1 provides a list of the *IPFM* files installed on your system, along with their directory paths.

Table B-1: Files Installed

Directory Path	File Name
/usr/opt/IPFM/bin/diag:	
	aptest
	ipfm_proc_action
	ipfm_proc_action.shutdown
	ipfm_test
	user_clr_crit
	user_clr_major
	user_clr_minor
	user_get_event
	user_req_alarm
	user_req_state
	user_req_status
	user_req_version
	user_set_active
	user_set_crit
	user_set_major
	user_set_minor
	user_set_out_of_service
	user_set_standby
	user_set_unavailable

Directory Path	File Name
/usr/opt/IPFM/bin/online:	
	config.script
	event_det
	event_man
	event_shutdown
	ipfm.conf
	ipfm_adm.pid
	ipfm_adm_queue
	ipfm_display.lock
	ipfm_event.lock
	ipfm_event_det.pid
	ipfm_init.d
	ipfm_shutdown
	pfm_sys_event_det.pid
	ln.script
	menu
	snmpd.conf
	system_event_det
/usr/opt/IPFM/doc	
	release.notes
/usr/opt/IPFM/src/api:	
	so_locations
/usr/opt/IPFM/src/apps:	
	ipfm_proc_action.c
	ln.script
	makefile
	mv.script
	user_clr_crit.c
	user_clr_major.c
	user_clr_minor.c
	user_req_alarm.c
	user_req_state.c
	user_req_status.c
	user_req_version.c
	user_set_active.c
	user_set_crit.c
	user_set_major.c
	user_set_minor.c
	user_set_out_of_service.c
	user_set_standby.c
	user_set_unavailable.c

Directory Path	File Name
/usr/opt/IPFM/src/aptest:	
	bld.aptest
	ln.script
	makefile
	mv.script
	udi.c
/usr/opt/IPFM/src/event_man:	
	ipfm_mib.my
	rfc1442.my
	v2-tc.my
/usr/opt/IPFM/src/include:	
	ipfm_alarm.h
/usr/sys/io/IPAP100:	
	files
	ipap.mod
	sysconfigtab

C
Configuration File

The configuration file used by the Event Manager (event_man) and Event Detector (`event_det`) processes during their initialization is shown below. A # sign character indicates a comment that should not be included in parameter configurations.

Some parameters are shared by the two processes, while others are exclusively for one process or the other. Parameters that enable *IPFM* hardware fault detection are shared by the processes; i.e. TEMP-CPU, FAN-CPU, POWER-CPU etc. are shared. The parameter POLLING is for the exclusive use of the Event Detector process to determine how often disk access will be checked. The parameters INDICATOR, ASC-LED and KEEP-ALIVE inform the Event Manager process that there will be an INDICATOR module (alarm indicator panel) on which to display an ASC-LED value and maintain a KEEP-ALIVE timer at the rate defined.

```
##############################################################################
# ipfm.conf                                                                  #
##############################################################################
#  Copyright (c) Digital Equipment Corporation, 1999. All Rights Reserved.   #
#  Unpublished rights reserved under the copyright laws of the United States. #
#                                                                            #
#  The software contained on this media is proprietary to and embodies the   #
#  confidential technology of Digital Equipment Corporation. Possession, use, #
#  duplication or dissemination of the software and media is authorized only  #
#  pursuant to a valid written license from Digital Equipment Corporation.   #
#                                                                            #
#  RESTRICTED RIGHTS LEGEND   Use, duplication, or disclosure by the U.S.    #
#  Government is subject to restrictions as set forth in Subparagraph        #
#  (c)(1)(ii) of DFARS 252.227-7013, or in FAR 52.227-19, as applicable.     #
##############################################################################
#
# Comments are prefixed by a '#' and continue for the remainder of the line
# on which they are encountered.
#
# ============================================================================
#
# Assignments for the Control and Indicator Modules.
# o present = TRUE,
# o not present = FALSE
#
# NOTE: For setld's scp to change the following parameters:
#       - there must be two tabs between CONTROL and its parameter and
#       - there must be one tab  between INDICATOR and its parameter.
```

```
#
CONTROL        TRUE                # TRUE if module present, FALSE if not.
INDICATOR      TRUE                # Can be TRUE only if CONTROL is TRUE.
#
# Indicator Module Parameters
#
ASC-LED         "\|/-\|/-\|/-\|/-"  # Pattern displayed in Status Display
KEEP-ALIVE      2                   # Number of seconds to update keepalive
SHUTOFF_SW      FALSE               # Detect Audible Disable Switch
AUDIBLE         TRUE                # Audible TRUE=enabled, FALSE=disabled
#
# ========================================================================
# Polling interval section   (in seconds - default 5)
#
POLLING     5
#
# ========================================================================
#
# Non-threshold events
#
# client configuration grammar:
# -----------------------------
#   Category of event
#   Severity of event
#   User definable event description and severity
#
# EVENT CATEGORY        SEVERITY    EVENT DESCRIPTION
# --------------        --------    -----------------------------------------
CPU                     # The SYSTEM event categories do not have SEVERITY
DISK                    # and EVENT DESCRIPTION fields.   The severity and
NETWORK                 # text are taken from the binary errorlog
MEMORY
#
#  System Environmental Events Section
#
TEMP-CPU                CRITICAL    "The CPU temperature is too high"
FAN-CPU                 MAJOR       "The CPU FAN has failed"
POWER-CPU               MAJOR       "The CPU redundent Power Supply has failed"
#
#  Expansion Chassis Events Section
#
#TEMP-EXT                CRITICAL    "The EXPANSION BOX temperature is too high"
#FAN-EXT                 MAJOR       "The EXPANSION BOX FAN has failed"
#POWER-EXT               CRITICAL    "The EXPANSION BOX Power Supply has failed"
#
#
# The following events have user definable text strings.   These events come
# commented out, but
# default to the event descriptions for the -48V power inverter.
# If you have the system (cabinet) power supply sensors attached to the power
# supply, uncomment out these event descriptions.
# If you wish to use the three external contacts for another purpose, the
# event text can be customized to provide a description of the event.
#
```

```
# NOTE: By default the following are commented out. For setld's scp to
#       change them there should no white space between the # sign and
#       the parameter.
#
#USER-EVENT1            CRITICAL   "The -48Vdc Inverter Fail relay on"
#USER-EVENT2            CRITICAL   "The -48Vdc Inverter Minor Alarm relay on"
#USER-EVENT3            CRITICAL   "The -48Vdc Inverter Major Alarm relay on"
#
# ------------------------------------------------------------------------
#
# Threshold events
#
# Line parameters:
#   Category of event (File System = FS)
#   File system mount point
#   Usage threshold level for severity
#   Severity
#
# File System Usage Threshold Section
# Format:
# FS    <filesystem mount point>  <usage threshold in bytes or %>   <severity>
# Examples:
# Monitor the root partition for usage less than 10%
# FS     /      10%    MAJOR
#
# Monitor the /usr partition for usage less than 20,000 bytes
# FS     /usr   20000  CRITICAL
#
FS      /        5%     MINOR
FS      /        2%     MAJOR
FS      /usr     10%    MINOR
FS      /usr     2%     MAJOR
# ------------------------------------------------------------------------
#
# Monitored processes events
#
# Line parameters:
#   Category of event (Process = PROCESS)
#   Process name
#   Process UID
#   Process count
#   Severity
#   Script to run (optional)
#
# Monitored Processes Section
# Format:
# PROCESS <process name>  <uid>  <count>  <severity>  [<script>]
# Example:
# Monitor for the ipfm event action process (ipfm_proc_action) with command to restart
# PROCESS ipfm_proc_action  0  1  CRITICAL  /usr/opt/IPFM/bin/diag/ipfm_proc_action
#
# ------------------------------------------------------------------------
#
```

```
# Alarm filtering threshold
#
# Line parameters:
#   Category of event (Alarm Filter = TRAP-THRESHOLD)
#   Alarm Filtering threshold
#
# Format:
# TRAP-THRESHOLD  <alarm filtering threshold severity>
#
# Examples:
# Set the alarm filtering threshold to MAJOR
# (only generate CRITICAL and MAJOR alarms)
#
# TRAP-THRESHOLD  MAJOR
# ========================================================================
```

D
SNMP Management Information Base

IPFM Private MIB Extension

The *IPFM* private MIB is shown below in Abstract Syntax Notation One (ASN.1) format:

```
   IPFM-MIB DEFINITIONS ::= BEGIN

   --  IPFM - IP Fault Manager MIB extension

   IMPORTS
           enterprises
                   FROM RFC1155-SMI
           DisplayString
                   FROM RFC1213-MIB
           OBJECT-TYPE
                   FROM RFC-1212;

   --  This MIB Module uses the extended OBJECT-TYPE macro as
   --  defined in [9].

   Dec           OBJECT IDENTIFIER ::= { enterprises 36 }
   ema           OBJECT IDENTIFIER ::= { dec 2 }
   sysobjectids  OBJECT IDENTIFIER ::= { ema 15 }
   ipfm          OBJECT IDENTIFIER ::= { sysobjectids 25 }

-- MIB Data Types

TrapSeverity ::= INTEGER {
    critical   (1),
    major      (2),
    minor      (3),
    warning    (4),
    clear      (5)
}

TrapProbableCause ::= INTEGER {
    adapterError                          (1),
    applicationSubsystemFailure           (2),
    bandwidthReduced                      (3),
    callEstablishmentError                (4),
    communicationsProtocolError           (5),
    communicationsSubsystemFailure        (6),
```

```
configurationOrCustomizationError          (7),
congestion                                 (8),
corruptData                                (9),
cpuCyclesLimitExceeded                     (10),
dataSetOrModemProblem                      (11),
degradedSignal                             (12),
dTE-DCEInterfaceError                      (13),
enclosureDoorOpen                          (14),
equipmentMalfunction                       (15),
excessiveVibration                         (16),
fileError                                  (17),
fireDetected                               (18),
floodDetected                              (19),
framingError                               (20),
heatingOrVentilationOrCoolingSystemProblem (21),
humidityUnacceptable                       (22),
inputOutputDeviceError                     (23),
inputDeviceError                           (24),
lANError                                   (25),
leakDetected                               (26),
localNodeTransmissionError                 (27),
lossOfFrame                                (28),
lossOfSignal                               (29),
materialSupplyExausted                     (30),
multiplexerProblem                         (31),
outofMemory                                (32),
outputDeviceError                          (33),
performanceDegraded                        (34),
powerProblem                               (35),
pressureUnacceptable                       (36),
processorProblem                           (37),
pumpFailure                                (38),
queueSizeExceeded                          (39),
receiveFailure                             (40),
receiverFailure                            (41),
remoteNodeTransmissionError                (42),
resourceAtOrNearingCapacity                (43),
responseTimeExcessive                      (44),
retransmissionRateExcessive                (45),
softwareError                              (46),
softwareProgramAbnormallyTerminated        (47),
softwareProgramError                       (48),
storageCapacityProblem                     (49),
temperatureUnacceptable                    (50),
thresholdCrossed                           (51),
timingProblem                              (52),
toxicLeakDetected                          (53),
transmitFailure                            (54),
transmitterFailure                         (55),
underlyingResourceUnavailable              (56),
versionMismatch                            (57),
snmpTrapColdStart                          (58),
snmpTrapWarmStart                          (59),
snmpTrapLinkDown                           (60),
```

```
    snmpTrapLinkUp                              (61),
    snmpTrapAuthenticationFailure               (62),
    snmpTrapEgpNeighborloss                     (63),
    snmpTrapEnterpriseSpecific                  (64),
    snmpTrapLinkUpDown                          (65)
}

TrapEventType ::= INTEGER {
    attributeValueChange             (1),
    communicationsAlarm              (2),
    environmentalAlarm               (3),
    equipmentAlarm                   (4),
    integrityVioloation              (5),
    objectCreation                   (6),
    objectDeletion                   (7),
    operationalVioloation            (8),
    physicalViolation                (9),
    processingErrorAlarm             (10),
    qualityofServiceAlarm            (11),
    relationshipChange               (12),
    securityServiceorMechanismViolation     (13),
    stateChange                      (14),
    timeDomainViolation              (15)
}

-- All alarms will have an ASCII discription associated
-- with them.

  AlarmType   ::= DisplayString

-- General information about this product and alarms.

  ipfmProductID OBJECT-TYPE
              SYNTAX  OBJECT IDENTIFIER
              ACCESS  read-only
              STATUS  mandatory
              DESCRIPTION
                      "The value of this object is the registered
                      object identifier for this product.  A value
                      of (0,0) should be returned if unsupported."
             ::= { ipfm 1 }

   nodeDescr OBJECT-TYPE
              SYNTAX  DisplayString
              ACCESS  read-only
              STATUS  mandatory
              DESCRIPTION
                      "A textual description of the particular node
                      within the IP system."
              ::= { ipfm 2 }
```

```
  nodeNumAlarms OBJECT-TYPE
             SYNTAX  INTEGER
             ACCESS  read-only
             STATUS  mandatory
             DESCRIPTION
                     "The current number of alarms on this particular
                     IP node."
             ::= { ipfm 3 }

  nodeStatus OBJECT-TYPE
             SYNTAX  DisplayString
             ACCESS  read-only
             STATUS  mandatory
             DESCRIPTION
                     "The alarm condition status of this node."
             ::= { ipfm 4 }

-- The Alarm Table.  There is one entry in the table
-- for each alarm recorded for each node in the IP system.

  alarmTable OBJECT-TYPE
      SYNTAX  SEQUENCE OF AlarmEntry
      ACCESS  not-accessible
      STATUS  mandatory
      DESCRIPTION
             "The conceptual table of alarms for all nodes which
              live in this IP system."
      ::= { ipfm 5 }

  alarmEntry OBJECT-TYPE
      SYNTAX  AlarmEntry
      ACCESS  not-accessible
      STATUS  mandatory
      DESCRIPTION
             "A conceptual entry for each alarm recorded for the
              particular node."
      INDEX { alarmIndex }
      ::= { alarmTable 1 }

-- the alarm entries may need additional fields added depending
-- on the info we want to monitor with them

  AlarmEntry
      ::= SEQUENCE {
          alarmIndex      INTEGER,
          alarmSeverity          AlarmDesc,
          alarmDescript          AlarmDesc,
          alarmStatus            INTEGER,
          alarmTimestamp         AlarmDesc,
          alarmIntSeverity    TrapSeverity,
          alarmProbableCause  TrapProbableCause,
          alarmEventType         TrapEventType
          }
```

```
alarmIndex OBJECT-TYPE
           SYNTAX  INTEGER
           ACCESS  read-only
           STATUS  mandatory
           DESCRIPTION
                   "A unique value for each AlarmEntry.  This
                   object may be set during row addition only.
                   Values of this object range from 1 to
                   NodeNumAlarms."
           ::= { alarmEntry 1 }

alarmSeverity OBJECT-TYPE
           SYNTAX  AlarmType
           ACCESS  read-only
           STATUS  mandatory
           DESCRIPTION
                  "The severity of the particular alarm CRITICAL,
                   MAJOR, MINOR, WARNING or INFO in ASCII text."
           ::= { alarmEntry 2 }

alarmDescript OBJECT-TYPE
           SYNTAX  AlarmType
           ACCESS  read-only
           STATUS  mandatory
           DESCRIPTION
                  "The ASCII test descriptive notation describing
                   the particular alarm."
           ::= { alarmEntry 3 }

alarmStatus  OBJECT-TYPE
           SYNTAX  DisplayString
           ACCESS  read-write
           STATUS  mandatory
           DESCRIPTION
                  "Status of this alarmEntry.  May be set to a
                   value of Delete only."
           ::= { alarmEntry 4 }

alarmTimestamp OBJECT-TYPE
           SYNTAX  AlarmType
           ACCESS  read-only
           STATUS  mandatory
           DESCRIPTION
                  "The ASCII test descriptive notation containing
                   the timestamp of this particular alarm event."
           ::= { alarmEntry 5 }

alarmIntSeverity OBJECT-TYPE
           SYNTAX TrapSeverity
           ACCESS read-only
           STATUS  mandatory
           DESCRIPTION
                 "The integer severity associated with the IPFM trap."
           ::= { alarmEntry 6 }
```

```
alarmProbableCause OBJECT-TYPE
            SYNTAX TrapProbableCause
            ACCESS read-only
            STATUS  mandatory
            DESCRIPTION
               "The probable cause associated with the IPFM trap."
            ::= { alarmEntry 7 }

alarmEventType OBJECT-TYPE
            SYNTAX TrapEventType
            ACCESS read-only
            STATUS  mandatory
            DESCRIPTION
                "The Event type associated with the IPFM trap."
            ::= { alarmEntry 8 }

nodeClrAllAlarms OBJECT-TYPE
            SYNTAX  INTEGER
            ACCESS  read-write
            STATUS  mandatory
            DESCRIPTION
                    "Variable to set to non-zero value to
                    clear all alarms in the IPFM event database"
            ::= { ipfm 6 }

nodeAckAllAlarms OBJECT-TYPE
            SYNTAX  INTEGER
            ACCESS  read-write
            STATUS  mandatory
            DESCRIPTION
                    "Variable to set to non-zero value to
                    acknowledge all alarms in the IPFM event database"
            ::= { ipfm 7 }

nodeRepostAllAlarms OBJECT-TYPE
            SYNTAX  INTEGER
            ACCESS  read-write
            STATUS  mandatory
            DESCRIPTION
                    "Variable to set to non-zero value to
                    repost all alarms in the IPFM event database"
            ::= { ipfm 8 }
```

```
ipfmCritTrap TRAP-TYPE
          ENTERPRISE ipfm
          VARIABLES {
            alarmIndex,
            alarmSeverity,
            alarmDescript,
            alarmStatus }
          DESCRIPTION
                 "This is the IPFM critical alarm trap, it
                  contains the severity, description, and
                  status of a critical alarm condition."
          ::= 199

ipfmMajorTrap TRAP-TYPE
          ENTERPRISE ipfm
          VARIABLES {
            alarmIndex,
            alarmSeverity,
            alarmDescript,
            alarmStatus }
          DESCRIPTION
                 "This is the IPFM major alarm trap, it
                  contains the severity, description, and
                  status of a major alarm condition."
          ::= 200

ipfmMinorTrap TRAP-TYPE
          ENTERPRISE ipfm
          VARIABLES {
            alarmIndex,
            alarmSeverity,
            alarmDescript,
            alarmStatus }
          DESCRIPTION
                 "This is the IPFM minor alarm trap, it
                  contains the severity, description, and
                  status of a minor alarm condition."
          ::= 201

ipfmWarningTrap TRAP-TYPE
          ENTERPRISE ipfm
          VARIABLES {
            alarmIndex,
            alarmSeverity,
            alarmDescript,
            alarmStatus }
          DESCRIPTION
                 "This is the IPFM alarm warning trap, it
                  contains the severity, description, and
                  status of an alarm condition which has
                  changed its status."
          ::= 202
```

```
                ipfmInfoTrap TRAP-TYPE
                        ENTERPRISE ipfm
                        VARIABLES {
                           alarmIndex,
                           alarmSeverity,
                           alarmDescript,
                           alarmStatus }
                        DESCRIPTION
                               "This is the IPFM alarm information trap, it
                                contains the severity, description, and
                                status of an alarm condition which has
                                changed its status."
                        ::= 203

                  ipfmNMSTrap TRAP-TYPE
                        ENTERPRISE ipfm
                        VARIABLES {
                           alarmIndex,
                           alarmSeverity,
                           alarmDescript,
                           alarmStatus,
                           alarmIntSeverity,
                           alarmProbableCause,
                           alarmEventType }
                       DESCRIPTION
                           "IPFM alarm for NMS auto correlation
                            MANAGEDOBJECT-DEFAULT: ipfm
                            SEVERITY-VARIABLE: alarmIntSeverity
                            NOTIFICATIONID-VARIABLE: alarmIndex
                                  PCAUSE-DEFAULT: 0
                                  PCAUSE-VARIABLE: alarmProbableCause
                                  EVENTTYPE-DEFAULT: 10
                                  EVENTTYPE-VARIABLE: alarmEventType
                            ADDITIONAL-TEXT-VARIABLES: { alarmDescript }"
                        ::= 204

END
```

E
Operation of the aptest Utility

The Alarm Panel Test (`aptest`) provides the *IPFM* operator (diagnostic user) with access to the *IPFM* control and indicator modules. When the operator starts this process, a prompt is displayed for the alarm control module to be accessed, as shown below.

```
cd /usr/opt/IPFM/bin/diag/aptest
aptest

Access Control Module number (0-7)? [0]
```

The prompt question is asked in order to support an off-line board test configuration where several modules can appear on the PCI bus at one time. (See "IPAP Device Driver" in Chapter 3 for more information.) The default answer to this question is shown in brackets ([0]), that is, unit 0. If a carriage return is typed, `aptest` continues by attempting to open a connection to the `ipap` driver using the online default control module `0'. If this attempt fails, an error message is displayed. Otherwise, the `aptest` main menu is displayed.

Items in the aptest Main Menu

Figure E-1 shows the `aptest` main menu items. The menu contains a list of tests (items 1-11), driver functions (items 12-18), an exerciser (item 19), and a monitor (item 20).

Figure E-1: aptest Menu Selections

```
0. Exit
1. Test Status register (LED set to A)
2. Test Enable Register (LED set to lB)
3. Test Control Register (LED set to C)
4. Test Interrupts (LED set to D)
5. Test Access to Indicator Module (LED set to E)
6. Test Critical Alarms (LED set to F)
7. Test Major Alarms (LED set to G)
8. Test Minor Alarms (LED set to H)
9. Test function that resets all visual alarms (LED set` to I)
10. Test disable audible alarm switch (LED set to J)
11. Test keep-alive function (LED set to K)
12. Reset hardware
13. Request driver to perform its startup
14. Request driver To Read Register
15. Request driver to Clear Register Bit(s)
```

```
16. Request driver to Set Register Bit(s)
17. Request driver to Get Interrupt Event
18. Request driver to Abort waiting for an Interrupt Event
19. Exercise Alarm Subsystem (LED set to A, B, C, E & L)
20. Monitor Alarm Subsystem Changes
    Select Operation [0]: 1
    Sorry, cannot execute while the Event Manager is running. <CR>
```

Also shown in Figure E-1 is the operator selection 1 and the resulting error message. Similar error messages are displayed for other selections if either the Event Detector or Alarm Database Manager process will be affected by the selection. In most cases when an error message is displayed, the operator must type a carriage return (<CR>) or press the Enter key before proceeding, in order to ensure that the message has been viewed.

Test Status Register

"Test Status Register" operates in a manner similar to "Test Enable Register," "Test Control Register," and "Test Access to Indicator Module." "Test Status Register" attempts to set the status display on the alarm indicator panel to "A" and put the alarm control module into diagnostic mode. In diagnostic mode, the Status Register is writeable as well as readable.

While in diagnostic mode, the test writes a one bit to every usable bit position in the Status Register, one bit at a time. It follows each write with a read and compares what has been read to what was written. If an error should occur, the data that is written and read back are displayed in hexadecimal format. The hexadecimal values of the bits written are listed below, along with their associated bit positions.

0x0001 bit 0	0x0010 bit 4	0x0200 bit 9
0x0002 bit 1	0x0020 bit 5	0x0400 bit 10
0x0004 bit 2	0x0040 bit 6	0x0800 bit 11
0x0008 bit 3	0x0080 bit 7	0x1000 bit 12

In addition, two alternate ones and zeros patterns (0x1555 and 0xAAA) are written to the Status Register following the sliding one pattern shown above.

If an error is encountered during the test, a message is displayed and the *IPFM* Operator is given the option of continuing the test or exiting. Continuing to run the test may further define the nature of a problem. An example error message is shown below:

```
Error - wrote 100 and read 0, continue test? [N]Y
```

Typing "Y" continues the test using the next bit pattern in the sequence listed above. Entering a carriage return, or "N" and a carriage return, aborts the test, and the usual completion message includes the error detected, as shown below.

```
Test completed with 1 data error(s) detected. Type <CR> to continue.
```

This test, when activated by selecting item 1 from the main menu, runs one pass of all the bit patterns shown above. However, when this test is called as a result of menu item 19 (Exercise Alarm Subsystem) being selected, multiple passes are run along with several other tests. In this case the error messages shown above do not appear. Instead, the exerciser's error count will reflect any errors occurring from this test as well as other tests called by the exerciser.

Test Enable Register

"Test Enable Register" operates in a manner similar to "Test Status Register," "Test Control Register," and "Test Access to Indicator." "Test Enable Register" attempts to set the status display on the alarm indicator panel to "B" and put the alarm control module into diagnostic mode. In diagnostic mode, the Enable Register is writeable as well as readable.

While in diagnostic mode, the test writes a one bit to every usable bit position in the Enable Register, one bit at a time. It follows each write with a read and compares what has been read to what was written. If an error should occur, the data that is written and read back are displayed in hexadecimal format. The hexadecimal values of the bits written are listed below, along with their associated bit position.

0x0001	bit 0	0x0040	bit 6
0x0002	bit 1	0x0080	bit 7
0x0004	bit 2	0x0100	bit 8
0x0008	bit 3	0x0200	bit 9
0x0010	bit 4	0x0400	bit 10
0x0020	bit 5	0x0800	bit 11

In addition, two alternate ones and zeros patterns (0x555 and 0xAAA) are written to the Enable Register following the sliding one pattern shown above.

If an error is encountered during the test, a message is displayed and the *IPFM* Operator is given the option of continuing the test or exiting. Continuing to run the test may further define the nature of a problem. Entering a carriage return, or "N" and carriage return, aborts the test. Typing "Y" continues the test using the next bit pattern in the sequence, as shown below.

```
Error - wrote 1 and read 3, continue test? [N]Y  y

Error - wrote 2 and read 3, continue test? [N]Y  y

Test completed with 2 data error(s) detected. Type <CR> to continue.
```

The Enable Register test, when activated by selecting item 2 from the main menu, runs one pass of all the bit patterns shown above. But, when this test is called as a result of menu item 19 (Exercise Alarm Subsystem) being selected, multiple passes are run along with several other tests. In this case, the error messages shown above do not appear. Instead, the exerciser's error count will reflect any errors occurring from this test as well as other tests called by the exerciser.

Test Control Register

"Test Control Register" operates in a manner similar to "Test Status Register," "Test Enable Register," and "Test Access to Indicator Module." The test attempts to set the status display on the alarm indicator panel to "C" and put the alarm control module into diagnostic mode. In diagnostic mode, the Control Register bits 0 through 16 are writeable as well as readable, while bit 17 is left to reset diagnostic mode.

While in diagnostic mode, the test writes a one bit to every usable bit position in the Control Register, one bit at a time. It follows each write with a read and compares what has been read to what was written. If an error should occur, both the data written and read back are displayed in hexadecimal format. The hexadecimal values of the bits written are listed below, along with their associated bit positions.

0x00001	bit 0	0x00200	bit 9
0x00002	bit 1	0x00400	bit 10
0x00004	bit 2	0x00800	bit 11
0x00008	bit 3	0x01000	bit 12
0x00010	bit 4	0x02000	bit 13
0x00020	bit 5	0x04000	bit 14
0x00040	bit 6	0x08000	bit 15
0x00080	bit 7	0x10000	bit 16
0x00100	bit 8		

In addition, two alternate ones and zeros patterns (0x5555 and 0xAAAA) are written to the Control Register following the sliding one pattern shown above.

If an error is encountered during the test, a message is displayed and the *IPFM* Operator is given the option of continuing the test or exiting. Continuing to run the test may further define the nature of a problem. An example of this kind of error message is shown below.

```
Error - wrote 20 and read 0, continue test? [N]Y
```

Typing "Y" continues the test using the next bit pattern in the sequence listed above. Entering a carriage return, or "N" and carriage return, aborts the test, and the usual completion message includes the error detected, as shown below.

```
Test completed with 1 data error(s) detected. Type <CR> to continue.
```

This test, when activated by selecting item 3 from the main menu, runs one pass of all the bit patterns shown above. However, when this test is called as a result of menu item 19 (Exercise Alarm Subsystem) being selected, multiple passes are run along with several other tests. In this case, the error messages shown above do not appear. Instead, the exerciser's error count will reflect any errors occurring from this test as well as other tests called by the exerciser.

Test Interrupts

"Test Interrupts" provides a means to verify that interrupts can be generated by the alarm control module and detected by the `ipap` driver. The actual verification is visually performed by the diagnostic user. The status display on the indicator panel is set to "d" while the test is running.

During this test, the diagnostic creates a background thread that requests the `ipap` driver to report interrupt events. In the foreground, the diagnostic requests the `ipap` driver to set the "Test Interrupt" bit of the Control Register, and then it waits two to four seconds. It is expected that the background thread will be returned by the driver during the wait and will display the reason for its return. If it does not return, an abort is issued to force the return.

The first one or two lines displayed to the user indicate whether or not interrupts are operational. The following message line indicates success. If this message occurs, it will be followed by the contents of the Status Register.

```
IPAP_GET_EVENT ioctl returned due to interrupt
```

A failure results in the following message being displayed.

```
IPAP_GET_EVENT ioctl returned with status…
status - ioctl was aborted
```

Test Access to Indicator Module

"Test Access to Indicator Module" operates in a manner similar to "Test Status Register," "Test Enable Register," and "Test Control Register." The test attempts to set the status display on the alarm indicator panel to "E" and writes a one bit to every data bit of the data path between the Control Register and alarm indicator panel, one bit at a time. It follows each write with a read and compares what has been read to what was written. If an error should occur, the data that is written and read back are displayed in hexadecimal format. The hexadecimal values of the bits written are listed below, along with their associated bit positions.

0x0001	bit 0
0x0002	bit 1
0x0004	bit 2
0x0008	bit 3
0x0010	bit 4
0x0020	bit 5
0x0040	bit 6

In addition, two alternate ones and zeros patterns (0x55 and 0x2A) are written to the CI bus following the sliding one pattern shown above.

If an error is encountered during the test, a message is displayed and the *IPFM* Operator is given the option of continuing the test or exiting. Continuing to run the test may further define the nature of a problem. An example error message is shown below.

```
Error - wrote 20 and read 0, continue test? [N]Y
```

Typing "Y" continues the test using the next bit pattern in the sequence listed above. Entering a carriage return or "N" and carriage return aborts the test, and the usual completion message includes the error detected, as shown below.

```
Test completed with 1 data error(s) detected. Type <CR> to continue.
```

This test, when activated by selecting item 5 from the main menu, runs one pass of all the bit patterns shown above. However, when this test is called as a result of menu item 19 (Exercise Alarm Subsystem) being selected, multiple passes are run, along with several other tests. In this case, the error message shown above does not appear. Instead, the exerciser's error count will reflect any error occurring from this test as well as other tests called by the exerciser.

Test Critical Alarms

While "Test Critical Alarms" is being executed, the status display on the alarm indicator panel is set to "F."

This test provides a means to verify that a critical audible and visual alarm can occur. The actual verification must be performed by the diagnostic user observing the Indicator Panel. When this test is started, it displays the following message.

```
Type <CR> to clear alarm
```

After observing that the alarm indicators are functioning, the diagnostic user can enter a carriage return to abort the audible and visual alarms.

Test Major Alarms

While "Test Major Alarms" is being executed, the status display on the alarm indicator panel is set to "G."

This test provides a means to verify that a major audible and visual alarm can occur. The actual verification must be performed by the diagnostic user observing the alarm indicator panel.

When this test is started it displays the following message.

```
Type <CR> to clear alarm
```

After observing that the alarm indicators are functioning, the diagnostic user can enter a carriage return to abort the audible and visual alarms.

Test Minor Alarms

While" Test Minor Alarms" is being executed the status display on the alarm indicator panel is set to "H."

This test provides a means to verify that a minor audible and visual alarm can occur. The actual verification must be performed by the diagnostic user observing the alarm indicator panel.

When this test is started it displays the following message.

```
Type <CR> to clear alarm
```

After observing that the alarm indicators are functioning, the diagnostic user can enter a carriage return to abort the audible and visual alarms.

Test Function that Resets All Alarms

While “Test Function that Resets All Alarms” is being executed, the status display on the alarm indicator panel is set to “I.”

This test provides a means to verify that the Control Register function that resets all alarms is functioning. The test sets the critical LED, major LED, minor LED, and critical audible indicators. Then it displays the following message.

```
Type <CR> to clear all alarm indicators
```

After observing that the alarm indicators are functioning, the diagnostic user can enter a carriage return to execute the function that aborts all audible and visual alarms. If this does not clear the alarms, the diagnostic user can use menu item 12 (Reset Hardware) to clear the alarms.

Test Disable Audible Alarm Switch

While “Test disable audible alarm switch” is being executed, the status display on the alarm indicator panel is set to “J.”

This test provides a means to verify that the disable audible alarm switch on the alarm indicator panel is functioning. The test sets the critical LED and critical audible indicators, then displays the following message.

```
Use the disable audible alarm switch to clear audible alarm then type <CR>
```

After using the switch and observing that only the critical audible alarm indicator has been cleared, the diagnostic user can enter a carriage return to abort the visual alarm. This action will also abort the critical audible alarm if the switch failed to do so.

Test Keep-Alive Function

“Test Keep-Alive Function” provides a means to verify that the hardware keep-alive function is operational. When started, it performs the following functions:

- Sets the status display on the alarm indicator panel to “K”
- Enables the keep-alive function
- Creates a background thread to maintain (reset) the hardware keep-alive counter at a default one-second rate
- Asks the user for the rate at which to maintain the timer, as shown below:

```
Enter Keep-Alive reset rate (1-15 sec) or just <CR> to abort the test.
```

If the user enters a value from 1 to 15 seconds, the keep-alive counter will be updated at that rate by the background thread. Values greater than 10 seconds should cause the timer to expire and set off an audible alarm. If the diagnostic user enters a carriage return, the alarm should be reset because the keep-alive function is disabled.

Reset Hardware

The Reset Hardware function uses the Control Register to:

- disable keep-alive timing
- disable critical audible and visual alarms
- disable major audible and visual alarms
- disable minor audible and visual alarms

The disabling of alarms is performed individually, that is, it is not done by using the hardware function, which resets all alarms.

Request Driver to Perform its Startup

"Request Driver to Perform its Startup" (item 13) results in a Startup `ioctl` call to the `ipap` driver. The Startup `ioctl` call is normally issued by the Alarm Database Manager to initialize the driver for on-line operation. The return from this call contains a basic indication as to whether or not the hardware is accessible.

If the `ipap` driver could not access the hardware during probe or attach time, then a status error message is displayed after the startup `ioctl` call has returned. Error messages are displayed in two lines as shown below.

```
IPAP_STARTUP ioctl failed
Error - Control Module Failed
```

An example of a successful startup completion message is shown below. This message includes the contents of the Status Register at attach time. The register contents are displayed as a hexadecimal value followed by a bit level breakdown. Bit 12 of the Status Register indicates the status of the alarm indicator panel attached to the alarm control module.

```
Status register contains the hexadecimal value   0


bit 0   = 0  Expansion chassis power okay
bit 1   = 0  Expansion chassis temperature okay
bit 2   = 0  Expansion chassis fan okay
bit 3   = 0  user input 1 okay
bit 4   = 0  user input 2 okay
bit 5   = 0  user input 3 okay
bit 6   = 0  user input 4 okay
bit 7   = 0  user input 5 okay
bit 8   = 0  user input 6 okay
bit 9   = 0  user input 7 okay
bit 10  = 0  user input 8 okay
bit 11  = 0  disable audible alarm not depressed
bit 12  = 0  Indicator Module okay
```

One of the tasks of the startup function is to force an interrupt. This action provides the initial value of the Status Register to the Event Manager process. If item 17 of the main menu ("Request Driver to Get Interrupt Event") was selected prior to item 13 ("Request Driver to Perform its Startup"), the contents of the Status Register are displayed twice. The second display is performed by the "Get Interrupt Event" thread.

Request Driver to Read Register

"Request Driver to Read Register" (item 14) results in a Read Register `ioctl` call to the `ipap` driver to obtain contents of one of three registers or the alarm indicator panel cable. The alarm indicator panel cable consists of the lines connecting the alarm control module and alarm indicator panel. The submenu shown below is used to select what is to be read.

1. `Status Register`
2. `Enable Register`
3. `Control Register`
4. `Cable Connection`
5. `Configuration Header`
6. `Interrupt CSR`

```
    Select Register [none]
```

The diagnostic user can abort the operation at this point by entering a carriage return. Typing the numbers one through four has the following results:

- 1 selects the Status Register for the read operation. The Status Register is normally a read-only register. The value returned is displayed in two forms, that is, as a hexadecimal value and as a bit value with the value of the bits interpreted in the display.
- 2 selects the Enable Register for the read operation. The Enable Register is normally a write-only register, and the returned value is meaningful only in diagnostic mode. In any case, a value is displayed in hexadecimal format.
- 3 selects the Control Register for the read operation. The Control Register is normally a write-only register, and the returned value is meaningful only in diagnostic mode. In any case, a value is displayed in hexadecimal format.
- 4 selects the alarm indicator panel cable for the read operation. The alarm indicator panel cable is both readable and writeable if the alarm indicator panel is attached. The value returned is displayed in hexadecimal format.

Request Driver to Clear Register

"Request Driver to Clear Register" (item 15) results in a Clear Register Bits `ioctl` call to the `ipap` driver to rewrite a register removing the bits specified in the call parameter. Two requests are made of the diagnostic user to obtain the parameters needed for the clear operation. The first, a sub-menu shown below, is displayed for the diagnostic user to select the target of the operation. The second request is used to obtain the bits to be removed from the register. The prompt includes "`0x`" to indicate that a hexadecimal value is being requested.

1. `Status Register`
2. `Enable Register`
3. `Control Register`
4. `Cable Connection`

```
    Select Register [none]  2

    Register Bit Value 0x
```

The diagnostic user can abort the operation when a target is request by entering a carriage return. The clear function has meaning for each target in normal on-line mode and diagnostic mode. Typing the numbers one through four has the following results:

- 1 selects the Status Register for the clear bits operation. The Status Register is normally a read-only register, and this function only has meaning in diagnostic mode. When the driver receives this request, it reads the current value of the Status Register. The driver assumes that the alarm control module is in diagnostic mode. It then removes the bits input, has a call parameter from the bits read, and writes the results back to the Status Register.
- 2 selects the Enable Register for the clear bits operation. The Enable Register is normally a write-only register, but the driver keeps a copy of the lasts bits written and operates on this copy to produce a value that is written to the Enable Register.
- 3 selects the Control Register for the clear bits operation. The Control Register is normally a write-only register, which results in the Control Register being written to all zeros, regardless of the input bits parameter value. The all-zero value is only meaningful to the Control Register in diagnostic mode where bits are both readable and writeable.
- 4 selects the alarm indicator panel cable for the clear bits operation. The alarm indicator panel cable is both readable and writeable. In online operation, the bits read are only useful in determining the last command issued to the alarm indicator panel. In test mode, the bits read are used to determine if the alarm indicator panel is accessible over a certain line. The clear bits function is useful in test mode to set the alarm indicator panel cable to a known state.

Request Driver to Set Register Bit(s)

"Request Driver to Set Register Bit(s)" (item 16) results in a Set Register Bits `ioctl` call to the `ipap` driver to rewrite a register with the bits specified in the call parameter. Two requests are made of the diagnostic user to obtain the parameters needed for the bit set operation. The first, a submenu shown below, is displayed for the diagnostic user to select the target of the operation. The second request is used to obtain the bits to be written to the register. The prompt includes "`0x`" to indicate that a hexadecimal value is being requested.

1. `Status Register`
2. `Enable Register`
3. `Control Register`
4. `Cable Connection`

```
Select Register [none]  2

Register Bit Value 0x
```

The diagnostic user can abort the operation when a target is request by entering a carriage return. The set bits function has meaning for each target in normal on-line mode and in diagnostic mode as described below. Typing the numbers one through four has the following results:

- 1 selects the Status Register for the set bits operation. The Status Register is normally a read-only register, and this function has meaning only in diagnostic mode. When the driver receives this request, it reads the current value of the Status Register. The driver assumes that the alarm control module is in diagnostic mode.

  It then inserts the bits input, has a call parameter into the bits read, and writes the results back to the Status Register.
- 2 selects the Enable Register for the set bits operation. The Enable Register is normally a write-only register, but the driver keeps a copy of the lasts bits written and operates on this copy to produce a value that is written to the Enable Register.

- 3 selects the Control Register for the clear bits operation. The Control Register is normally a write-only register, which results in the Control Register being written to with the bits input regardless of what was previously in the register.
- 4 selects the alarm indicator panel cable for the bit set operation. The alarm indicator panel cable is both readable and writeable. In online operations, the bits read are only useful in determining the last command issued to the alarm indicator panel. In test mode, the bits read are used to determine whether the alarm indicator panel is accessible over a certain line. The bit set function is useful in test mode to set an alarm indicator panel cable line to be tested.

Request Driver to Get interrupt Event

"Request Driver to Get Interrupt Event" (item 17) results in a background thread being created. This thread posts a request to the `ipap` driver for notification of an interrupt event. When this selection is made, an immediate display of an interrupt event may take place or the request may just remain outstanding, depending on whether or not an event is waiting to be reported. If there is no immediate report, the diagnostic user is allowed to use the `aptest` to perform other functions, including generation of an interrupt event. This can be done by setting the "Test Interrupt" bit (0x8000) in the Control Register.

An example of a reported interrupt event appears below.

```
IPAP_GET_EVENT ioctl returned due to interrupt

Status register contains the hexadecimal value      1000

bit 0   = 0  Expansion chassis power okay
bit 1   = 0  Expansion chassis temperature okay
bit 2   = 0  Expansion chassis fan okay
bit 3   = 0  user input 1 okay
bit 4   = 0  user input 2 okay
bit 5   = 0  user input 3 okay
bit 6   = 0  user input 4 okay
bit 7   = 0  user input 5 okay
bit 8   = 0  user input 6 okay
bit 9   = 0  user input 7 okay
bit 10 = 0  user input 8 okay
bit 11 = 0  disable audible alarm not depressed
bit 12 = 1  Indicator Module okay
```

Request Driver to Abort Waiting for an Interrupt Event

"Request Driver to Abort Waiting for an Interrupt Event" (item 18) results in a request to the `ipap` driver for the notification of an interrupt event to be aborted. If such a request were outstanding, it would be returned and would display the following message.

```
IPAP_GET_EVENT ioctl returned with…
status -ioctl was aborted
```

Exercise Alarm Subsystem

"Exercise Alarm Subsystem" (item 19) results in the `aptest` continuously accessing the hardware to perform the tests mentioned in items 1, 2, 3, and 5 of the main menu. This test is the only one that can access multiple module sets. In addition, the alarm indicators selected by the diagnostic user are set and cleared. To access multiple module sets, enter the highest unit number when prompted at the startup utility. The diagnostic user is prompted for alarm parameters as shown in the example below.

1. `Exit` (this can be typed at any time during the test)
2. `Set and Clear Both Audible and Visual Alarms,`
3. `Set and Clear Only Visual Alarms,`
4. `Set and Clear Only Audible Alarms`

```
Select one of the above.  2

How long should each alarm indication remain on 1-60 seconds?  [2]  5

Test started on Tue Mar    3 10:33:37  EST 1997 (typing  0 aborts the
test)   0

Test was aborted Tue Mar   3 10:35:00  EST 1997 with 0 errors
detected

Waiting for test thread to exit.

Type <CR> to continue
```

As indicated earlier, typing "0" aborts the test. Two parameters were selected in the above example. The first parameter set up the test so that only the visual alarms for critical, major, and minor alarms were used during the test. The second parameter set up a delay value for each alarm level before it is removed and the next alarm level is set. The other tests are run serially with the alarm indications test and are therefore delayed by the amount of time the alarms are left on.

Monitor Alarm Subsystem Changes

"Monitor Alarm Subsystem Changes" (item 20) results in the `aptest` requesting continuously (once a second) information about the state of the alarm subsystem. Information returned from the driver is displayed only if a change occurs in any of the previous values returned. The initial request always appears as a change and is displayed as shown in the following example.

```
Event Detector Opened count                 = 1
Event Manager Opened count                  = 1
User Diagnostic Interface Opened count      = 1 <- last change

Information about unit 0;

Startup Call Status Value ---------         SUCCESS
Startup Status Register Bits ------            1000
Current Event Return Status -------        SLEEPING
Current Event status register bits             1000
Data last written to Cable --------              6f
Data last written to Enable Reg --              3e8
Data last written to Control Reg -              100
Interrupt in progress indicator ---               0
Current state of  critical visual ---           OFF
Current state of  critical audible -            OFF
Current state of  major visual ----              ON
Current state of  major audible ---              ON
Current state of  minor visual  ----            OFF
Current state of  minor audible ---             OFF

Type <CR> to abort Monitoring Alarm Sub-system Changes
```

The following subsections can help interpret call status and register bit definitions displayed by the "Monitor Alarm Subsystem" menu selection.

Call Status Definitions

Call status is displayed as “Startup Call Status Value” and “Current Event Return Status.” The “Startup Call Status Value” represents the status of the alarm subsystem when the dynamic `ipap` driver is loaded into the system. At this time there is an attempt to access the hardware. This value will not be changed until it is unloaded and reinstalled. “Current Event Return Status” is the status value stored in the driver’s Get Event `ioctl` status location and can change from one moment to the next. Table E–1 provides call status definitions.

Table E–1: Call Status Definitions

Status	Definition
NGPAR	Bad or unsupported parameter for this configuration
NOTUSED	Unit not initialized for use
SUCCESS	General success status
ABORTED	Get event operation was aborted by application
CLOSED	The connection between the driver and Event Detector is being closed
EVENT	There is an interrupt event to be reported
NOIOCTL	There is no event, and the Event Detector is not waiting for an event to occur
SLEEPING	There is no event, and the Event Detector is waiting for an event to occur
NGIND	Indicator module is not operational
NGCTRL	The `ipap` driver cannot access the alarm control module

Alarm Indicator Panel Cable Data Values

Some values written to the alarm indicator panel cable that connects the alarm control module and alarm indicator panel result in the status display displaying an ASC character. The hexadecimal values and the characters they produce are listed in Table E–2.

Table E–2: Indicator Module Cable Data Values

Hex, ASC	Hex, ASC	Hex, ASC	Hex, ASC	Hex, ASC	Hex, ASC
20	30 0	40 @	50 P	60 `	70 p
21 !	31 1	41 A	51 Q	61 a	71 q
22 “	32 2	42 B	52 R	62 b	72 r
23 #	33 3	43 C	53 S	63 c	73 s
24 $	34 4	44 D	54 T	64 d	74 t
25 %	35 5	45 E	55 U	65 e	75 u
26 &	36 6	46 F	56 V	66 f	76 v
27 ‘	37 7	47 G	57 W	67 g	77 w
28 (	38 8	48 H	58 X	68 h	78 x
29)	39 9	49 I	59 Y	69 i	79 y
2A *	3A :	4A J	5A Z	6A j	7A z
2B +	3B ;	4B K	5B [	6B k	7B {
2C ,	3C <	4C L	5C \	6C l	7C |
2D -	3D =	4D M	5D]	6D m	7D }
2E .	3C >	4E N	5E ^	6E n	7E ~
2F /	3D ?	4F O	5F _	6F o	7F

Status Register Bit Definitions

The off (0) and on (1) states of Status Register bits are described in Table E–3.

Table E–3: Status Register Bit Definitions

Bit	Off	Meaning
0	0	Expansion chassis power okay
1	0	Expansion chassis temperature okay
2	0	Expansion chassis fan okay
3	0	User input 1 okay
4	0	User input 2 okay
5	0	User input 3 okay
6	0	User input 4 okay
7	0	User input 5 okay
8	0	User input 6 okay
9	0	User input 7 okay
10	0	User input 8 okay
11	0	Disable Audible Alarm Switch not depressed
12	0	Indicator Module failed

Bit	On	Meaning
0	1	Expansion chassis power out of range
1	1	Expansion chassis temperature out of range
2	1	Expansion chassis fan out of range
3	1	User input 1 failed
4	1	User input 2 failed
5	1	User input 3 failed
6	1	User input 4 failed
7	1	User input 5 failed
8	1	User input 6 failed
9	1	User input 7 failed
10	1	User input 8 failed
11	1	Disable Audible Alarm Switch depressed
12	1	Indicator Module okay

Enable Register Bit Definitions

The off (0) and on (1) state of Enable Register bits are described in Table E–4. During online operation these bits are set by the Event Manager by their definition in the `ipfm.conf` file.

Table E–4: Enable Register Bit Definitions

Bit	On	Meaning
0	1	Expansion chassis power enabled for detection
1	1	Expansion chassis temperature enabled for detection
2	1	Expansion chassis fan enabled for detection
3	1	User input 1 enabled for detection
4	1	User input 2 enabled for detection
5	1	User input 3 enabled for detection
6	1	User input 4 enabled for detection
7	1	User input 5 enabled for detection
8	1	User input 6 enabled for detection
9	1	User input 7 enabled for detection
10	1	User input 8 enabled for detection
11	1	Disable Audible Alarm Switch enabled for detection
12	1	Indicator Module enabled for status

Control Register Bit Definition

The meanings of the set state (value of 1) of Control Register bits are listed in Table E–5.

Table E–5: Control Register Bit Definitions

Bit	Meaning
0	Critical alarm (used with bits 3, 4, 5 & 6 to define alarm type & operation)
1	Major alarm (used with bits 3, 4, 5 & 6 to define alarm type & operation)
2	Minor alarm (used with bits 3, 4, 5 & 6 to define alarm type & operation)
3	Turn on alarm LED
4	Turn on audible alarm
5	Turn off alarm LED
6	Turn off audible alarm
7	Turn off all indicators
8	Enable interrupts
9	Acknowledge interrupt in progress
10	Disable interrupts
11	Clear Status Register
12	Enable Keep-alive timer
13	Disable Keep-alive timer
14	Reset Keep-alive timer count to keep it running
15	Perform a test interrupt
16	Place the Alarm-subsystem in diagnostic mode (Register Read/Write)
17	Disable diagnostic mode

F
Hardware Reference

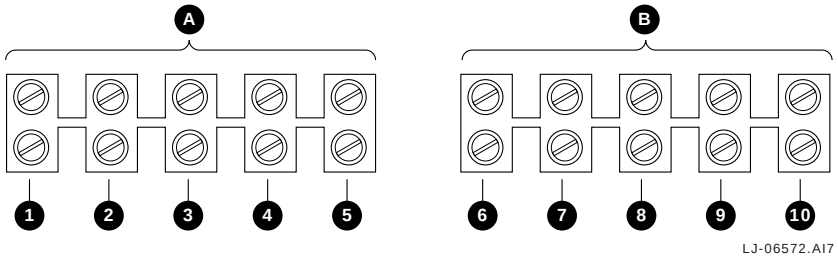
This appendix describes following components of the *AlphaServer Intelligent Peripheral (IP) Platform* on which the *Intelligent Peripheral Fault Manager* is installed:

- Dry contacts
- Alarm input wiring

Dry Contacts

The dry contacts terminal connectors (see Figure F–1) are located on the rear panel of the alarm display unit. The four contacts on the input terminal block correspond to Critical, Major, Minor, and Audio Shut-off alarm inputs. The output terminal block contacts correspond to Critical, Major, Minor, and Audio Shut-off alarm outputs. These terminal blocks can be used to connect remote alarm indicators. When an alarm condition occurs, the corresponding relay is closed to connect a voltage on the input terminal to a remote alarming device on the output terminal. See Table F–1 for the dry contact relay functional specification. Additionally, an audible alarm can be remotely disabled by connecting to the Shut-off Alarm contacts. Table F–2 provides the dry contact electrical specifications.

Figure F-1: Dry Contact Terminal Connectors



A	Input Terminal Block	**B**	Output Terminal Block
❶	Ext. Critical Alarm	❻	Critical Alarm
❷	Ext. Major Alarm	❼	Major Alarm
❸	Ext. Minor Alarm	❽	Minor Alarm
❹	Ext. Shut-off Alarm	❾	Shut-off Alarm
❺	Ground	❿	Ground

Table F–1: Dry Contact Relay Functional Specification

		Relay State		
Power	**Condition**	**Critical**	**Major**	**Minor**
On	No alarm	Open	Open	Open
	Critical	Closed	Open	Open
	Minor	Open	Open	Closed
	Major	Open	Closed	Open
Off	Battery OK	Open	Open	Closed
	Battery NOK	Closed	Closed	Closed

Dry Contact Specifications

Table F–2 contains the rating, input, and output specifications for the dry contact terminals.

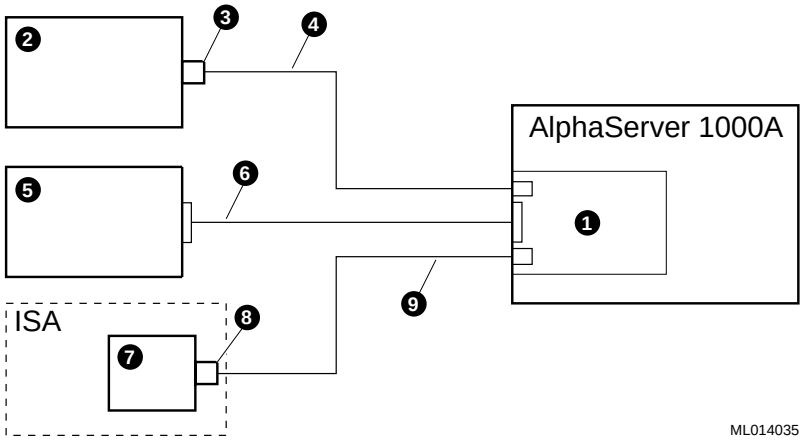
Table F–2: Dry Contact Specifications

Material and Ratings	
Contact material	Gold-clad silver
Rating (resistive)	
Maximum switching power	60 W, 125 VA
Maximum switching voltage	220 Vdc, 250 Vac
Maximum switching current	2 A DC, 2 A AC
Maximum carrying current	3 A DC, 3 A AC
Input Specifications	
Type	Relay coil
Rated input voltage	5 Vdc nominal
Maximum input	10 Vdc at 50 C
Pick-up voltage	3.5 Vdc maximum
Drop-out voltage	0.5 Vdc minimum
Coil resistance	125 ohms, +/- 10%
Output Specifications	
Contact rating (switching)	
Voltage	220 Vdc, 250 Vac (maximum)
Current	2 A DC, 2 A AC (maximum)
UL/CSA	0.6 A, 125 Vac
	0.6 A, 110 Vdc
	2.0 A, 30 Vdc
Coil rating	200 mW typically

Alarm Input Wiring

The following diagram of the Intelligent Peripheral (IP) duplex platform displays the alarm input wiring connections between the ISA bus expansion chassis, the *AlphaServer* 1000A processors, and the alarm indicator panel.

Figure F–2: Alarm Input Wiring Diagram



❶ Alarm control module installed in the *AlphaServer* 1000A system

❷ -48 Vdc power inverter

❸ D-sub 25-pin connector

❹ Inverter alarm cable

❺ Alarm indicator panel

❻ Alarm indicator panel cable

❼ Alarm sensor module in the ISA bus expansion chassis

❽ D-sub 9-pin connector

❾ ISA bus expansion chassis alarm cable

Wiring User Alarm Inputs

This section provides information for wiring user TTL signal alarm inputs to the 8-pin MJ connector on the alarm control module installed in the *AlphaServer* 1000A system.

Figure F–3 shows the 8-pin MJ connector on the alarm control module that is used for connecting user alarm inputs.

Figure F–3: Alarm Control Module 8-Pin MJ Connector

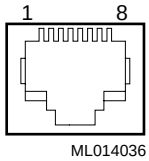


Table F–3 provides a pinout listing of the pins on the alarm control module 8-pin MJ connector. This table should be used to determine the correct wiring required for user alarm inputs.

Table F–3: Alarm Control Module 8-Pin MJ Connector Pinout

Pin No.	Signal
1	Inverter Fail or user input 1 Return
2	Inverter Fail or user input 1
3	Inverter Minor or user input 2 Return
4	Inverter Minor or user input 2
5	Inverter Major or user input 3 Return
6	Inverter Major or user input 3
7	Not used
8	Not used

Glossary

agent

A background task running on each monitored node, scanning devices and data structures and generating event messages in an internal list. The agent responds to requests for information by the network management station (NMS). The agent is responsible for performing get and set operations, generating traps, and controlling access.

alarm conditions

Alarm conditions occur when a component or process malfunctions.

alarm panel

LEDs visible through the front panel of the DTP chassis that indicate system status, alarm status and power supply.

alarm status

One of three alarm conditions, that is major, minor or critical alarms.

AlphaServer

Compaq's new generation of server systems based on the Alpha 64-bit computing architecture.

boot device

The device from which the system bootstrap software is acquired.

boot

Short for bootstrap. To load an operating system into memory.

bus

A collection of many transmission lines or wires. The bus interconnects computer system components, providing a communications path for addresses, data, and control information or external terminals and systems in a communications network.

CD-ROM

Compact disc read-only memory. The optical removable media used in a compact disc reader.

central processing unit (CPU)

The unit of the computer that is responsible for interpreting and executing instructions.

critical alarm condition

A severe condition that affects the performance of the IP platform; it requires immediate attention.

event

A problem or situation detected by the system.

initialization

The sequence of steps that prepare the computer system to start. Initialization occurs after a system has been powered up.

ISA

Industry Standard Architecture. An 8-bit or 16-bit industry-standard I/O bus, widely used in personal computer products. The EISA bus is a superset of the ISA bus.

light-emitting diode (LED)

An indicator of status on an IP (intelligent peripheral) subsystem.

major alarm condition

A serious disruption of service or malfunction of important circuits; it requires immediate attention.

mass storage device

An input/output device on which data is stored. Typical mass storage devices include disks, magnetic tapes, and CD-ROM.

minor alarm condition

An alarm condition that causes minimal disturbance in service.

module

A hardware or software component that is a self-contained system interacting with a larger system. Hardware modules are often made to plug into a main system.

network

A collection of computers, terminals, and other devices together with the hardware and software that enables them to exchange data and share resources over either short or long distances.

network management station (NMS)

A PC or workstation equipped with an Ethernet, FDDI, or Token Ring network module and *HUBwatch* software that enables it to communicate with and manage network modules.

PCI

Peripheral component interconnect. An industry-standard expansion I/O bus that is the preferred bus for high-performance I/O options. PCI is available in a 32-bit version and a 64-bit version.

polling interval

The amount of time between requests for event messages from the consolidate to the agents on the monitored nodes.

SCSI

Small Computer Systems Interface. An ANSI-standard interface for connecting disks and other peripheral devices to computer systems. Some devices are supported under the SCSI-1 specification; others are supported under the SCSI-2 specification.

server

A network node or specialized device that provides and manages access to shared network resources, such as hard disks, printers, and software.

standby

The status of the backup system in a redundant configuration, where the primary system is functioning normally.

StorageWorks

Compaq's modular storage subsystem (MSS), which is the core technology of the Alpha SCSI-2 mass storage solution. *StorageWorks* consists of a family of low-cost mass storage products that can be configured to meet current and future storage needs.

system disk

The device on which the operating system resides.

unavailable

The status of a troubled system or a system under repair. The maintenance center can initiate placing a system in and taking a system out of unavailable mode.

Index

P

S

U

W