High Availability for Linux Operating Systems

Demo Guide

September 2005

05-1860-004

Publication Date: September 2005

Document Number: 05-1860-004

Intel Converged Communications, Inc.
1515 Route 10
Parsippany, NJ 07054

For **Technical Support**, visit the Intel Telecom Support Resources website at:
http://developer.intel.com/design/telecom/support

For **Products and Services Information**, visit the Intel Telecom and Compute Products website at:
http://www.intel.com/design/network/products/telecom

For **Sales Offices** and other contact information, visit the Buy Telecom Products page at:
http://www.intel.com/buy/networking/telecom.htm

Contents

Tables

Revision History

This revision history summarizes the changes made in each published version of this document.

Document No.	Publication Date	Description of Revisions
05-1860-004	September 2005	Global change: Updated to include redundant host information.
05-1860-003	August 2005	Software Requirements section: HSK kernel installation/configuration information is now located in the system release Installation Guide.
05-1860-002	March 2005	Global change: Changed Redundant System Slot demo name to Redundant Host demo. Redundant Host Program Prerequisite section: Added note about Redundant Host steps required after initial installation of the Intel® Dialogic System Software. Running the Demo section: Changed Peripheral Fault Manager demo shortcut keys to T (stop a board), R (remove a board), S (start a board). Files Used by the Demo section: updated file names to reflect those of the current release.
05-1860-001	September 2002	Initial version of document.

About This Publication

The following topics provide information about this publication:

Purpose

This publication provides information about the set of sample programs that demonstrate how to integrate peripheral hot swap and redundant host into Intel® Dialogic® cPCI computer telephony systems.

The high availability demo is designed to illustrate how you can develop an application that coincides with the Intel Continuum of Availability, wherein you can expand system capacity without completely restructuring your application.

Note: Refer to the *Economics of High Availability: An Intel Primer* white paper located at *http://www.intel.com/network/csp/resources/white_papers/* for complete information about the various high availability architectures.

Intended Audience

This publication is written for the following audience:

- Distributors
- System Integrators
- Toolkit Developers
- Independent Software Vendors (ISVs)
- Value Added Resellers (VARs)
- Original Equipment Manufacturers (OEMs)

How to Use This Publication

Refer to this publication after you have installed the hardware and the Intel Dialogic System Software that includes the high availability demo programs.

This publication assumes that you are familiar with the following:

- redundant host and peripheral hot swap high availability architectures
- C and C++ programming languages
- computer telephony terms and concepts
- Linux* operating system

The information in this guide is organized as follows:

- Chapter 1, "Demo Description" provides a brief overview of the individual programs that comprise the high availability demo.
- Chapter 2, "System Requirements" discusses the hardware and software required to run the high availability demo programs.
- Chapter 3, "Preparing to Run the Demo" lists the procedures you must follow before running the high availability demo programs.
- Chapter 4, "Running the Demo" describes procedural information for starting, using and stopping the high availability demo programs.
- Chapter 5, "Demo Details" provides a table that lists the files used by the demo programs.

Related Information

Refer to the following documents and Web sites for more information about the software packages and procedures used to develop the high availability demo programs:

- *OA&M API for Linux Programming Guide*
- *OA&M API for Linux Library Reference*
- *System Release for Linux Administration Guide*
- *System Release for Linux Software Installation Guide*
- *Intel® DM3 Architecture Products on Linux Configuration Guide*
- *Global Call API Programming Guide*
- *Global Call API Library Reference*
- *System Release Guide*
- *System Release Update*
- *Economics of High Availability: An Intel Primer* white paper located at *http://www.intel.com/network/csp/resources/white_papers/*
- *http://developer.intel.com/design/telecom/support/* (for technical support)
- *http://www.intel.com/design/network/products/telecom/index.htm* (for product information)

Demo Description 1

This chapter provides a brief description of the programs that comprise the High Availability demo.

The High Availability demo illustrates how to build a scalable, highly-available application with Intel® Dialogic® System Software. When the demo programs are run in conjunction with one another, they support a system that is capable of the following:

- detection of peripheral board faults
- fault identification to determine the origin of faults so that the component can be diagnosed and corrected or replaced
- fault isolation to prevent the fault from affecting the rest of the system
- Single Board Computer (SBC) redundancy and peripheral board redundancy to restore the system to an operational state if a critical fault occurs
- dynamic scalability, allowing you to add new peripheral boards to increase system capacity

The demo consists of the following programs:

Revenue Generating Application/Peripheral Fault Manager Program Set

The Revenue Generating Application (RGA)/Peripheral Fault Manager (PFM) program set illustrates how to develop a highly available call control application using the following Intel Dialogic libraries:

- Global Call API Library
- OA&M API Library
- Standard Runtime API Library

The Revenue Generating Application uses the Global Call API library to make and receive calls with the Intel Netstructure® boards in your system. As the RGA makes and receives calls, the Peripheral Fault Manager (PFM) uses the OA&M API event notification framework (ADMIN_CHANNEL and FAULT_CHANNEL events) and the Power On Self Test-on-demand utility to provide the following features, all of which are done without stopping the RGA:

- monitoring peripheral boards for Control Processor (CP) and Signal Processor (SP) faults (fault detection)
- automatically running Power On Self Test (POST) diagnostics on any peripheral board that generates a CP or SP fault (fault identification and diagnosis)
- automatically restarting a peripheral board that passes POST diagnostics (fault recovery)
- prompting a system administrator to replace a peripheral board that fails POST diagnostics (fault isolation)
- basic hot swap of peripheral boards (fault repair)

Note: The Revenue Generating Application and the Peripheral Fault Manager must be run in tandem.

Redundant Host Program

The Redundant Host program is used in conjunction with the cPCI chassis vendor demo that is installed as part of the Intel Dialogic System Software. The Redundant Host program

demonstrates redundant Single Board Computer (SBC) support on cPCI systems. One of the SBCs operates in Active mode while the second (redundant) SBC operates in Standby mode. The chassis vendor's Redundant Host software is used to generate a TAKEOVER event, which simulates a system-critical fault on the Active SBC. The application notifies the Redundant Host program via the Pigeon Point Hot Swap APIs or the *dlgrhdemo* that a TAKEOVER event has occurred and automatically invokes the *takeover.sh* script that switches system control to the Standby SBC. The script loads the drivers of the Standby SBC, restarts all the Intel NetStructure boards and allows the computer telephony application to be restarted, with minimal application downtime.

Notes: ***1.*** The *takeover.sh* script can be customized according to your system/application. For example, you can edit the script to re-start the Revenue Generating Application HA demo program.

2. Refer to the chassis vendor documentation included in the **rh** or **rss** directory on the Intel Dialogic System Release CD-ROM for complete information about generating a TAKEOVER event.

Refer to the *Intel® NetStructure™ ZT 5550 High Availability Processor Board Software Manual* located in the **rss** directory on your Intel® Dialogic system release CompactPCI for Windows 2000 CD-ROM for complete information about generating a TAKEOVER event.

System Requirements 2

This chapter describes the requirements for running the various programs of the High Availability demo. Topics include:

2.1 Hardware Requirements

To run the High Availability demo programs, you need the following hardware:

- A cPCI chassis that supports peripheral hot swap and/or redundant host architectures

  Note: Refer to the *Release Guide* for a list of supported chassis.

- At least one Intel NetStructure® board

  Note: The Revenue Generating Application (RGA) supports a maximum of five boards. However, if there are both voice and network devices in the system, the ratio of network devices ("dti") to voice devices ("dxxx") for all boards used by the RGA must be less than or equal to five. Refer to the *Standard Runtime Library API Programming Guide* for a complete overview of network and voice devices.

- At least one T1 or E1 crossover cable

  Note: Crossover cables are only required to run the Revenue Generating Application/Peripheral Fault Manager program set.

2.2 Software Requirements

To run the High Availability demo programs, you need the Intel® Dialogic® System Release on Linux* Operating Systems, including the configured Hot Swap Kit (HSK) and Redundant Host kernel. Refer to the *Installation Guide* for information about configuring your system with the HSK Redundant Host kernel.

Preparing to Run the Demo 3

This chapter provides information about procedures to follow before running the High Availability demo programs. Topics include:

3.1 Connecting to External Equipment

You must connect each of the board's T1/E1 network interfaces with a crossover cable so that the board can operate in loopback mode. For example, if you have a DM/V960-4T1-cPCI board installed in your system, you must use a T1 crossover cable to connect the J1 network interface to the J2 network interface and use a second crossover cable to connect the J3 network interface to the J4 network interface. This allows the Revenue Generating Application to make calls on the channels of the J1 and J3 network interfaces and receive those same calls on the channels of the J2 and J4 network interfaces.

3.2 Editing Configuration Files

This section provides information about editing configuration files for the Revenue Generating Application program. The following topics are included:

- Editing the redundant.cfg File for Peripheral Board Redundancy
- Editing the .config File for the Revenue Generating Application

3.2.1 Editing the redundant.cfg File for Peripheral Board Redundancy

The Revenue Generating Application (RGA) allows you to optionally configure your system for peripheral board redundancy. Peripheral board redundancy allows you to assign a backup board for each board that is initially used by the RGA. If a board has a backup board defined and the initial board is taken out of the system due to a Hot Swap removal, the RGA automatically begins making and receiving calls on the backup board.

If you are configuring your system for peripheral board redundancy, you must set up the initial/backup board pairs by manually editing the *redundant.cfg* file (located at */usr/dialogic/demos/ha_demos/rgademo*).

The *redundant.cfg* file contains one section, **[PhysicalBoardRedundancy]**. The **[PhysicalBoardRedundancy]** section contains entries for assigning board pairs. Board pairs are assigned using the physical slot number of the boards installed in your system. For example, the following *redundant.cfg* file entry assigns no backup boards for the boards in physical slots 1, 4 and 5, but assigns the board installed in physical slot 3 as a backup for the board installed in physical slot 2:

```
[PhysicalBoardRedundancy]
BoardPair = 1, 0
BoardPair = 2, 3
BoardPair = 4, 0
BoardPair = 5, 0
```

Notes: ***1.*** An entry of 0 indicates that a board does not have a backup board assigned.

2. Entries in the *redundant.cfg* file must be formatted exactly as shown in the example, with a space before and after the =, and a space after each comma that separates the initial/backup board pairs.

3.2.2 Editing the .config File for the Revenue Generating Application

If you are running the Revenue Generating Application/Peripheral Fault Manager program set, you must edit the Intel NetStructure® board's *.config* file and generate a new *.fcd* file so that your board(s) can operate in loopback mode.

Refer to the *Intel® DM3 Architecture Products on Linux Configuration Guide* for complete information about *.config* file parameters, *.config* file naming conventions, editing the *.config* file and using the **fcdgen** utility to generate a new *.fcd* file.

Use the following procedure to configure boards used by the RGA to operate in loopback mode:

1. Determine the correct configuration file set (*.config*, *.pcd*, *.fcd*) for your board based on network protocol.

2. Locate the *.config* file in */usr/dialogic/data.*

3. Open the *.config* file with a text editor and change the **CCS_PROTOCOL_MODE (ISDN Protocol Mode)** parameter (number 0x17) to 1 for each network interface that is receiving calls via the crossover cable that was installed according to the instructions in Section 3.1, "Connecting to External Equipment", on page 13. For example, if you have a DM/V960-4T1-cPCI board installed in your system, and the J1 network interface is connected to the J2 network interface via a crossover cable, you must edit the *.config* file to ensure that the **CCS_PROTOCOL_MODE (ISDN Protocol Mode)** parameter for the J2, or second, network interface is set to 1 for NETWORK_MODE. Likewise, if the J3 network interface is connected to the J4 network interface via a second crossover cable, you must set the **CCS_PROTOCOL_MODE (ISDN Protocol Mode)** parameter for the J4, or fourth, network interface to 1 for NETWORK_MODE.

   The following example is taken from the *qs_isdn_qsigt1.config* file, note that the **CCS_PROTOCOL_MODE (ISDN Protocol Mode)** parameter has already been changed to 1 (NETWORK_MODE) for the second network interface [CCS.2] and the fourth network interface [CCS.4]:

```
[CCS.1]
! Q.931 Timer Values in milliseconds
Setparm=0x0b,4000   ! Q.931 timer 303. Default=4000 msec.
Setparm=0x0d,4000   ! Q.931 timer 305. Default=4000 msec.
Setparm=0x0e,4000   ! Q.931 timer 308. Default=4000 msec.
Setparm=0x0f,10000  ! Q.931 timer 310. Default=10000 msec.
Setparm=0x10,4000   ! Q.931 timer 313. Default=4000 msec.
Setparm=0x15,1000   ! TEI retry timer Default=1000 msec.
Setparm=0x16,900    ! TEI state 4 min stability time. Default=900 msec.

Setparm=0x13,0      ! Symmetrical C.R. protocol. 0=disable 1=enable
Setparm=0x18,0      ! Enable Feature Test
Setparm=0x17,0      ! ISDN Protocol Mode.  0 = USER_MODE; 1=NETWORK_MODE
Setparm=0x7,11        ! CCS_SWITCH_TYPE - QSIGE1 = 10, QSIGT1 = 11
Setparm=0x9,0         ! 0=disabled, 1=enable Layer 2 access.
                      ! When Layer 2 access is enabled call control is no longer
                      ! supported for the channels on this line.

[CCS.2]
! Q.931 Timer Values in milliseconds
Setparm=0x0b,4000   ! Q.931 timer 303. Default=4000 msec.
Setparm=0x0d,4000   ! Q.931 timer 305. Default=4000 msec.
Setparm=0x0e,4000   ! Q.931 timer 308. Default=4000 msec.
Setparm=0x0f,10000  ! Q.931 timer 310. Default=10000 msec.
Setparm=0x10,4000   ! Q.931 timer 313. Default=4000 msec.
Setparm=0x15,1000   ! TEI retry timer Default=1000 msec.
Setparm=0x16,900    ! TEI state 4 min stability time. Default=900 msec.

Setparm=0x13,0      ! Symmetrical C.R. protocol. 0=disable 1=enable
Setparm=0x18,0      ! Enable Feature Test
Setparm=0x17,1      ! ISDN Protocol Mode.  0 = USER_MODE; 1=NETWORK_MODE
Setparm=0x7,11        ! CCS_SWITCH_TYPE - QSIGE1 = 10, QSIGT1 = 11
Setparm=0x9,0         ! 0=disabled, 1=enable Layer 2 access.
                      ! When Layer 2 access is enabled call control is no longer
                      ! supported for the channels on this line.


[CCS.3]
! Q.931 Timer Values in milliseconds
Setparm=0x0b,4000   ! Q.931 timer 303. Default=4000 msec.
Setparm=0x0d,4000   ! Q.931 timer 305. Default=4000 msec.
Setparm=0x0e,4000   ! Q.931 timer 308. Default=4000 msec.
Setparm=0x0f,10000  ! Q.931 timer 310. Default=10000 msec.
Setparm=0x10,4000   ! Q.931 timer 313. Default=4000 msec.
Setparm=0x15,1000   ! TEI retry timer Default=1000 msec.
Setparm=0x16,900    ! TEI state 4 min stability time. Default=900 msec.

Setparm=0x13,0      ! Symmetrical C.R. protocol. 0=disable 1=enable
Setparm=0x18,0      ! Enable Feature Test
Setparm=0x17,0      ! ISDN Protocol Mode.  0 = USER_MODE; 1=NETWORK_MODE
Setparm=0x7,11        ! CCS_SWITCH_TYPE - QSIGE1 = 10, QSIGT1 = 11
Setparm=0x9,0         ! 0=disabled, 1=enable Layer 2 access.
                      ! When Layer 2 access is enabled call control is no longer
                      ! supported for the channels on this line.

[CCS.4]
! Q.931 Timer Values in milliseconds
Setparm=0x0b,4000   ! Q.931 timer 303. Default=4000 msec.
Setparm=0x0d,4000   ! Q.931 timer 305. Default=4000 msec.
Setparm=0x0e,4000   ! Q.931 timer 308. Default=4000 msec.
Setparm=0x0f,10000  ! Q.931 timer 310. Default=10000 msec.
Setparm=0x10,4000   ! Q.931 timer 313. Default=4000 msec.
Setparm=0x15,1000   ! TEI retry timer Default=1000 msec.
Setparm=0x16,900    ! TEI state 4 min stability time. Default=900 msec.

Setparm=0x13,0      ! Symmetrical C.R. protocol. 0=disable 1=enable
Setparm=0x18,0      ! Enable Feature Test
Setparm=0x17,1      ! ISDN Protocol Mode.  0 = USER_MODE; 1=NETWORK_MODE
Setparm=0x7,11        ! CCS_SWITCH_TYPE - QSIGE1 = 10, QSIGT1 = 11
Setparm=0x9,0         ! 0=disabled, 1=enable Layer 2 access.
                      ! When Layer 2 access is enabled call control is no longer
                      ! supported for the channels on this line.
```

4. Save and close the updated *.config* file.

5. Use the **fcdgen** utility to generate an updated *.fcd* file for your Intel NetStructure® board.

6. Download the *.pcd* file and the updated *.fcd* file to your board(s) and start the Intel® Dialogic System Services using the procedures outlined in the *Intel® DM3 Architecture Products on Linux Configuration Guide*.

3.3 Adjusting Monitor Display Size

For both the Revenue Generating Application and the Peripheral Fault Manager, the terminal display that the programs execute in must be set to at least 80x25. When you execute the programs, if the terminal window is smaller than the required size, you will receive the following message:

```
Your terminal must be minimally 80x25.
```

If this message is displayed, you must manually resize the terminal window.

3.4 Redundant Host Program Prerequisite

As a prerequisite to running the Redundant Host program, you must replace the startup scripts of both SBCs with a customized Intel Dialogic wrapper that directs the operating system to load the device drivers onto the Active SBC and start the TAKEOVER event monitor on the Standby SBC (the Intel Dialogic Redundant Host program will load the device drivers onto the Standby SBC when a TAKEOVER event occurs).

You must perform the following procedure before running the Redundant Host program:

1. Copy */usr/inti.d/ct_intel* to */usr/dialogic/init.d/.*
2. Replace the */etc/init.d/ct_inte*l script with the *startsystem.sh* script. The *startsystem.sh* script is the customized Intel Dialogic wrapper that enables the Redundant Host demo.
3. Reboot the system. After the system has been rebooted, the *startsystem.sh* script automatically starts the Intel® Dialogic System Service on the Active SBC and the TAKEOVER event monitor daemon on the Standby SBC.

Note: When the Intel® Dialogic System Software is installed for the *first time* on both SBCs, only the Active SBC will have the drivers installed and the boards enumerated. The Standby SBC will not have the boards enumerated in the system. To address this, you must prep the boards on the Standby SBC by performing a cooperative switchover and enabling the boards. After this is done, the Standby SBC can download and activate the boards when a the application notifies the Redundant Host demo that a failover/takeover event has occurred. This notification is done via the Pigeon Point Hot Swap APIs or the Pigeon Point *rhdemo* demo program.

Running the Demo 4

This chapter provides instructions on using the High Availability demo. Topics include:

4.1 Starting the Demo

If you are running the Revenue Generating Application/Peripheral Fault Manager program set, the Revenue Generating Application (RGA) must be started first. The RGA *rgademo* file located at */usr/dialogic/demos/ha_demos/rgademo*. When the RGA program is started, it immediately opens the channels on the installed boards and begins making and receiving calls. If a board has been designated as a backup board in the *redundant.cfg* file, it is displayed in STANDBY mode.

Note: Boards designated as backup boards do not begin making/receiving calls until the initial board in the board pair is physically removed from the system.

After the RGA begins making and receiving calls, you can then start the Peripheral Fault Manager (PFM) by running the *pfmdemo* file located at */usr/dialogic/demos/ha_demos/pfmdemo*. When the PFM is started, it displays a list of installed Intel NetStructure® boards according to their physical slot number.

The Redundant Host program is automatically started after you reboot the system according to the procedure outlined in Section 3.4, “Redundant Host Program Prerequisite”, on page 16.

4.2 Using the Demo

This section contains information about using the various programs that comprise the High Availability Demo. The following topic is included:

- Using the Revenue Generating Application/Peripheral Fault Manager Programs
- Using the Redundant Host Program

4.2.1 Using the Revenue Generating Application/Peripheral Fault Manager Programs

The Revenue Generating Application main window provides information about each board used by the application. The following information is displayed in the RGA main window:

- logical ID of each board (board number)
- unique Addressable Unit Identifier (AUID) of each board

- physical slot number of each board
- current status of the board
- number of calls processed by each board (number is updated every 100 calls)
- event notification framework events that are received by the Peripheral Fault Manager and passed to the RGA. The following event notification framework events may be displayed in the RGA main window:
  - DLGC_EVT_BLADE_STOPPED
  - DLGC_EVT_BLADE_REMOVED
  - DLGC_EVT_BLADE_DETECTED
  - DLGC_EVT_BLADE_STARTED

To stop processing calls on a board without using the PFM, access the RGA main window and enter ‘p’, followed by the board number of the board you wish to stop. This closes all open channels on the specified board.

You can refresh the display in the RGA main window at anytime by pressing ‘r’.

4.2.1.1 Diagnosing Faults with the Peripheral Fault Manager

The Peripheral Fault Manager monitors the event notification framework’s FAULT_CHANNEL for the following events:

DLGC_EVT_CP_FAILURE
: Generated when a Control Processor (CP) fault occurs on a board being used by the Revenue Generating Application.

DLGC_EVT_SP_FAILURE
: Generated when a Signal Processor (SP) fault occurs on a board being used by the Revenue Generating Application.

When the Peripheral Fault Manager detects a CP or SP fault event, the following routine is automatically invoked:

1. The PFM informs the RGA that a CP or SP fault has been generated by a board in the system.
2. The PFM instructs the RGA to quiesce the board. The RGA responds by stopping all calls on the board and displays the boards status as ‘quiesced’.
3. After the board has been ‘quiesced’ by the RGA, the PFM calls the `stopbrd` utility to stop the board. This generates a DLGC_EVT_BLADE_STOPPED event that is displayed in the RGA main window.
4. After the board has been successfully stopped, the PFM invokes the `diagbrd` utility to perform a Power On Self Test (POST) on the board that generated the CP/SP fault.
   - 4a. If the board passes the POST, the PFM display instructs the user to “Press s to start board”.
   - 4b. If the board fails the POST, the PFM automatically calls the `removebrd` utility to remove the board. When the `removebrd` utility successfully removes the board, the PFM displays

a “Failed Diagnostics. Remove board when blue light is lit” message. The system administrator can then physically remove the board when the Out of Service (Blue) LED lights.

4.2.1.2 Using the Peripheral Fault Manager to Replace a Board

The following procedure describes how to remove and replace (peripheral hot swap) a board that is being used by the Revenue Generating Application:

1. In the Peripheral Fault Manager main window, use the up/down arrow keys to highlight the board you wish to remove and replace.

2. Press t. The PFM informs the RGA of the stop board request. The RGA responds by updating the board’s status to ‘quiesced’, meaning it is no longer making and receiving calls.

3. After the board has been quiesced, the PFM automatically invokes the `stopbrd` utility to stop the board. When the board has been successfully stopped, the RGA displays the following information in its main window:

   ```
   Event Detected:
   DLGC_EVT_BLADE_STOPPED with auid X, Board Y, slot Z
   ```

   where *X* is the Addressable Unit Identifier (AUID) of the board that is being stopped, *Y* is the number of the board and *Z* is the slot number occupied by the board.

4. The PFM display is updated with the following text:

   ```
   "Press r for removal or s to start"
   ```

5. Press r. The PFM automatically calls the `removebrd` utility.

6. When the `removebrd` utility is complete, the PFM display is updated as follows:

   ```
   "Remove Board when Blue light is lit"
   ```

7. When the board’s Out of Service (Blue) LED lights, unlock the board extraction handles. The PFM display is updated as follows:

   ```
   "Handles unlocked"
   ```

8. Remove the board from the system. This generates a DLGC_EVT_BLADE_REMOVED event that is displayed in the RGA main window. The PFM display is updated as follows:

   ```
   "Board Removed"
   ```

9. Insert a board into the system. This generates a DLGC_EVT_BLADE_DETECTED event that is displayed in the RGA main window. The PFM display is updated with the following text:

   ```
   "Press s to start board"
   ```

10. Configure the board for use by the RGA according to the procedures outlined in Chapter 3, “Preparing to Run the Demo”.

11. Press s to start the inserted board. This generates a DLGC_EVT_BLADE_STARTED event that is displayed in the RGA main window and updates the PFM display as follows:

```
"Board ready for application"
```

12. The RGA creates an entry for the board in its main window and automatically begins making and receiving calls with the inserted board.

4.2.1.3 Using the Peripheral Fault Manager to Add a Board

The following procedure is used to add a board for use by the Revenue Generating Application (expand system capacity):

1. Pre-configure an empty slot according to the procedures outlined in the *Intel® DM3 Architecture Products on Linux Configuration Guide.*

2. Insert a board into the pre-configured slot. This generates a DLGC_EVT_BLADE_DETECTED event that is displayed in the RGA main window. The PFM display is updated with the following text:

   ```
   "Press s to start board"
   ```

3. Configure the board for use by the RGA according to the procedures outlined in Chapter 3, "Preparing to Run the Demo".

4. Press s to start the inserted board. This generates a DLGC_EVT_BLADE_STARTED event that is displayed in the RGA main window and updates the PFM display as follows:

   ```
   "Board ready for application"
   ```

5. The RGA then creates an entry for the board in its main window and automatically begins making and receiving calls with the added board.

4.2.1.4 Using the Peripheral Fault Manager to Remove a Board

The following procedure shows how to remove a board that is being used by the Revenue Generating Application:

1. In the Peripheral Fault Manager main window, use the up/down arrow keys to highlight the board you wish to remove and replace.

2. Press t. The PFM informs the RGA of the stop board request. The RGA responds by updating the board's status to 'quiesced', meaning it is no longer making and receiving calls.

3. After the board has been quiesced, the PFM automatically invokes the `stopbrd` utility to stop the board. When the board has been successfully stopped, the RGA displays the following information in its main window:

   ```
   Event Detected:
   DLGC_EVT_BLADE_STOPPED with auid X, Board Y, slot Z
   ```

   where *X* is the Addressable Unit Identifier (AUID) of the board that is being stopped, *Y* is the number of the board and *Z* is the slot number occupied by the board.

4. The PFM display is updated with the following text:

   ```
   "Press r for removal or s to start"
   ```

5. Press r. The PFM automatically calls the `removebrd` utility.

6. When the `removebrd` utility is complete, the PFM display is updated as follows:

   ```
   "Remove Board when Blue light is lit"
   ```

7. When the board's Out of Service (Blue) LED lights, unlock the board extraction handles. The PFM display is updated as follows:

   ```
   "Handles unlocked"
   ```

8. Remove the board from the system. This generates a DLGC_EVT_BLADE_REMOVED event that is displayed in the RGA main window. The PFM display is updated as follows:

   ```
   "Board Removed"
   ```

4.2.2 Using the Redundant Host Program

The Redundant Host program establishes a TAKEOVER event monitor on the Standby SBC. You must use the chassis vendor's Redundant Host software demo to initiate a TAKEOVER event from the command line. The TAKEOVER event simulates a fault on the Active SBC. Once a TAKEOVER event has been detected by the Standby SBC, the Redundant Host demo invokes the *takeover.sh* script, which loads the drivers onto the Standby SBC and calls the `dlstart` utility to restart the Intel® Dialogic® System Software. After `dlstart` has completely restarted the software, you can then manually restart the computer telephony application (assuming the *takeover.sh* script hasn't already been modified to start the application automatically).

Note: Refer to the cPCI chassis vendor's documentation included with the Intel Dialogic System Release on Linux* Operating Systems for complete information about generating TAKEOVER events. However, keep in mind that when a TAKEOVER event occurs, the SBC that went from Active to Standby must be rebooted before becoming Active again.

4.3 Stopping the Demo

To stop the RGA, press 'q' in the main window.To stop the Peripheral Fault Manager press 'q' within the PFM main window.

Demo Details 5

This chapter provides more details about the High Availability demo programs.

5.1 Files Used by the Demo

Table 1 lists the files used by the High Availability demo programs:

Table 1. Files Used by the High Availability Demo Programs

Directory	File Name	Purpose
usr/dialogic/demos/ha_demos/rgademo	rgademo	Revenue Generating Application executable
usr/dialogic/demos/ha_demos/rgademo	redundant.cfg	file used by the Revenue Generating Application to assign initial/backup pairs for peripheral board redundancy
usr/dialogic/demos/ha_demos/rgademo	mainThread.cpp	primary Revenue Generating Application source code file
usr/dialogic/demos/ha_demos/pfmdemo	pfmdemo	Peripheral Fault Manager executable
usr/dialogic/demos/ha_demos/pfmdemo	main.cpp	primary Peripheral Fault Manager source code file
usr/dialogic/demos/ha_demos/pfmdemo	adminhandler.h	defines and implements a class that is derived from the OA&M API CEventHandlerAdaptor class for handling events on the ADMIN_CHANNEL and the FAULT_CHANNEL
usr/dialogic/demos/ha_demos/pfmdemo	adminhandler.cpp	implements the CEventHandlerAdaptor::HandleEvent( ) function that is called when an ADMIN_CHANNEL event or a FAULT_CHANNEL event is received
usr/dialogic/demos/ha_demos/dlgrhdemo	main.cpp	primary Redundant Host source code file
usr/dialogic/demos/ha_demos/dlgrhdemo	startsystem.sh	script that starts the drivers and the application on the Active SBC
usr/dialogic/demos/ha_demos/dlgrhdemo	takeover.sh	script that is called by the Redundant Host demo when a TAKEOVER event occurs
usr/dialogic/demos/ha_demos/dlgrhdemo	IRHManager.cpp	defines an interface for a class that allows the main( ) routine to initialize the Redundant Host system
usr/dialogic/demos/ha_demos/dlgrhdemo	RHManager.h	defines the implementation of the chasssis vendor Redundant Host interface

Index

Z