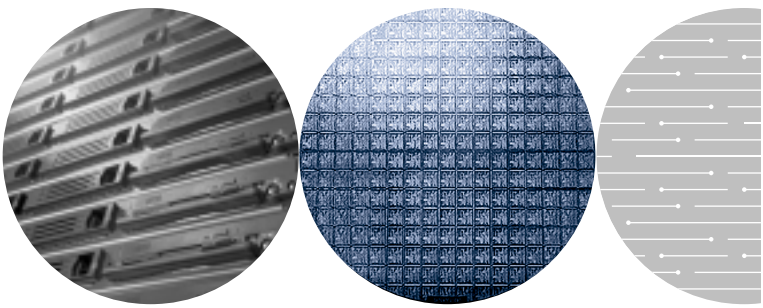
Application Note

High Availability Features in Intel® Dialogic® System Release 6.0 CompactPCI* for Windows*

Intel in Communications



Contents

Figures

Tables

Introduction

This paper discusses the new features of Intel® Dialogic® System Release 6.0 for CompactPCI that enable users to build high availability (HA) into telecommunication systems. The HA features included in the System Release are:

- Peripheral hot-swap (PHS)
- Redundant system slot (RSS)
- Peripheral redundancy

The System Release supports a range of CompactPCI servers and single-board computers. This paper discusses in detail the RSS on the Performance Technologies* ZT5084 platform and the Intel® NetStructure™ ZT5550 single-board computer.

Peripheral Hot-Swap

Peripheral hot-swap (PHS) for CompactPCI systems is one of the most popular and cost-effective HA architectures. It allows for the online repair, upgrade, or addition of peripherals in a CompactPCI chassis without the need to power down the system. Peripherals can be telephony boards, disk drives, fans, power supplies, management and alarm modules, and more. PHS can have a significant impact on reducing downtime, both planned and unplanned.

PHS, as defined by the PICMG 2.1 and PICMG 2.12 specs, can be divided into two models: basic hot-swap and full hot-swap.

Basic Hot-Swap

The basic hot-swap model defines the parameters and attributes to insert or remove a peripheral device (e.g., a board) without causing any interrupts or activity on the PCI bus. Since the backplane needs to be passive for this zero-activity, some operator intervention is required at the console to indicate to the operating system that a board needs to be removed or inserted. When instructed by the operator, the operating system shuts down all active operations on the board, thus making the board non-functional in the system and safe for removal. If a board is being inserted, the new CompactPCI signal, ENUM#, informs the operating system (OS) that a board is requesting enumeration and allocation of resources. This model is the simplest and less automatic.

Full Hot-Swap

The full hot-swap model enhances the basic hot-swap model by defining a method that-indicates to the OS that a board is being inserted into or removed from the system. This is accomplished by a microswitch attached to the IEEE 1101.10-compliant board that signals the OS that an operator is about to insert or remove a board. This microswitch is connected to the handles of the board that are used to either insert or remove the peripheral device. When the microswitch is tripped, the enumeration interrupt (ENUM#) indicates to the OS about the insertion or removal. The operating system then signals the operator, via a blue LED on the face of the board, that it is acceptable to remove the board. If a board were being inserted, the OS would automatically configure the board, thus eliminating the need for reconfiguring the system at the console. This model is more complex to implement, but does not require any operator intervention.

Redundant System Slot

Redundant system slot (RSS) systems provide for redundant, hot-swappable, single-board computers (SBCs) in a CompactPCI system. Such a system builds on the capabilities of a peripheral hot-swap (PHS) CompactPCI system by eliminating the SBC as a single point of failure.

An RSS platform can support different modes of operation such as Active-Standby mode and Active-Active mode. The two SBCs in the RSS platform are installed on the same CompactPCI backplane, which can be configured with software to simultaneously or independently control two CompactPCI bus segments.

In the Active-Standby mode, there are two SBCs in the RSS platform. However, only one SBC is active at any time. This one SBC has control of all the I/O Slots. The standby SBC is aware of — and to some extent synchronized with — operations on the active SBC and ready for failover to occur. The standby SBC monitors and takes over operations if a failure should occur on the active SBC.

High Availability Software Component Architecture

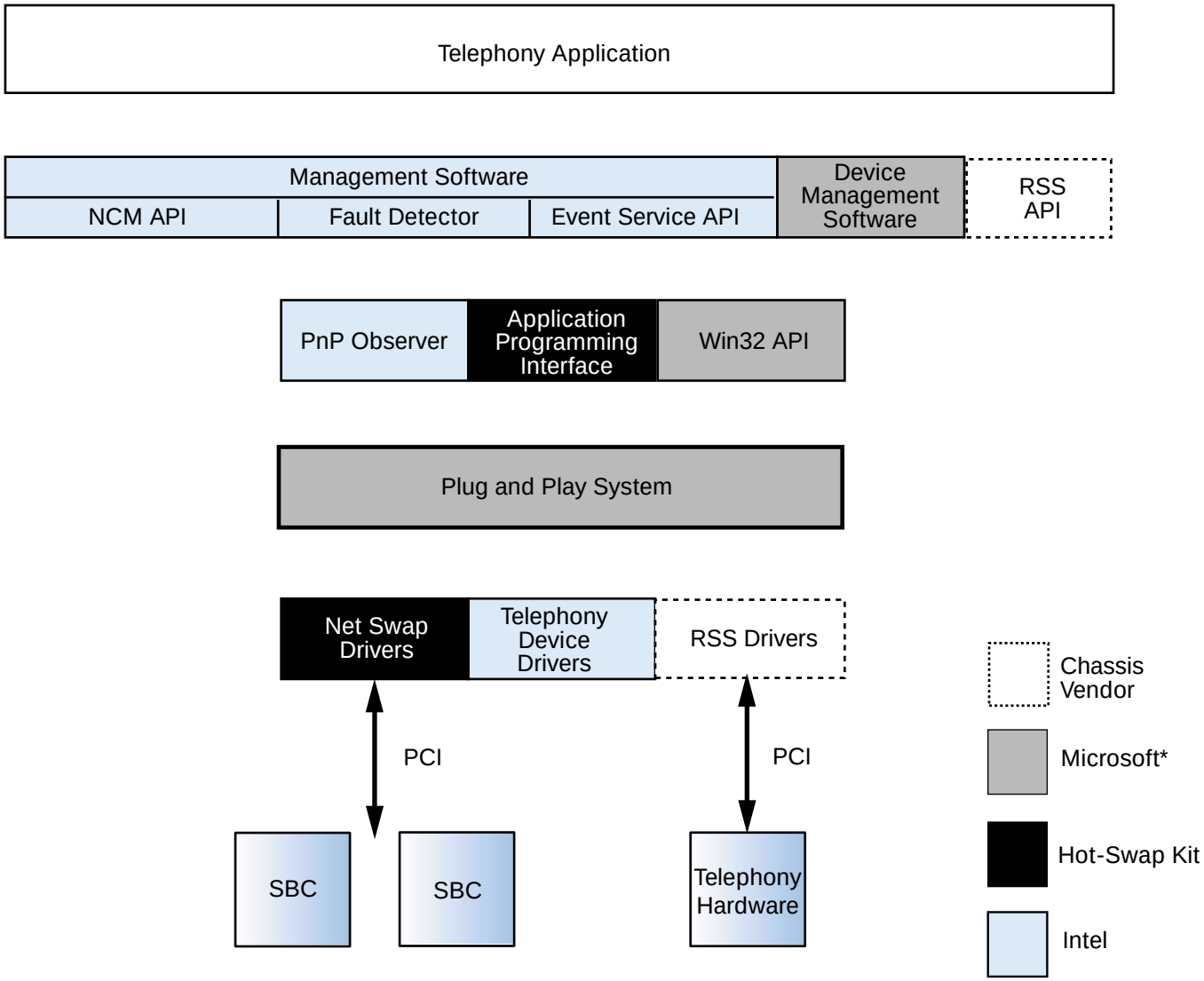


Figure 1: HA software architecture

In the Active-Active mode (also called the Split mode), each SBC controls one CompactPCI bus segment. Each SBC also acts as a standby for the bus segment it does not control. In this model, both SBCs are able to contribute resources. Customized software is provided to take advantage of this model's benefits, which include fast failover into an Active-Standby state and load sharing and redundancy (both of which can be achieved in this mode of operation). When an SBC fails, the second active SBC will takes over operations of the failed one and continues operations on the peripherals managed by the failed SBC.

There are two primary advantages of RSS:

1. **Elimination of the SBC as a single point of failure**, removing the need to duplicate costly peripherals and extensive application changes.
2. **"Dark closet" operation**, where there is no need to have an operator to add/remove malfunctioning peripheral devices.

Note that the RSS standard (PICMG 2.13) has not yet been ratified and many CompactPCI platform vendors currently offer proprietary and non-interoperable solutions. Thus, it is important to carefully evaluate the needs and requirements of the system being built and choose the right high-availability solution.

Scenario of Registering with the Event Notification Framework

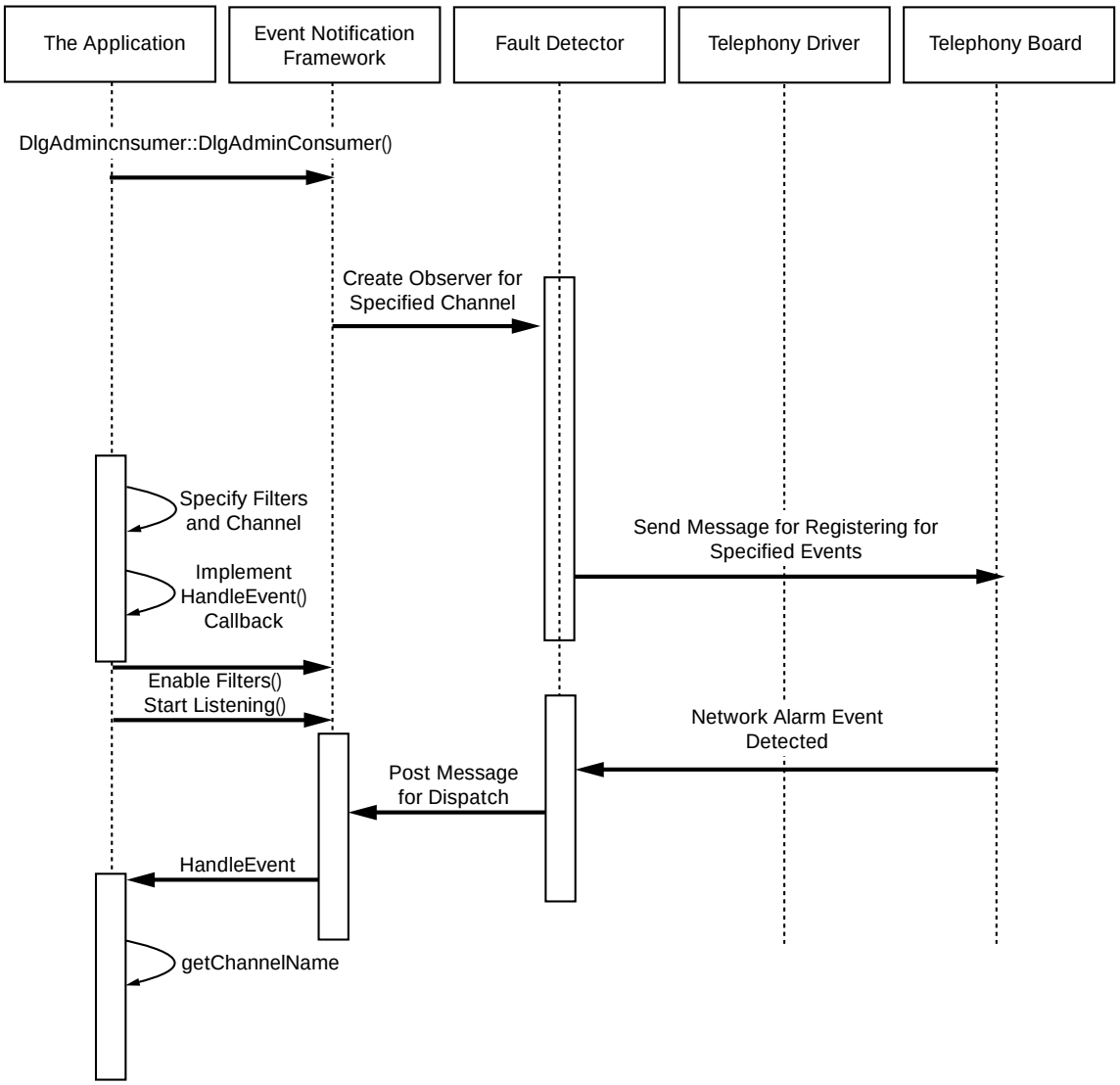


Figure 2: Example of registering events

Peripheral Redundancy

While PHS is effective in reducing the time to repair, by itself it does not protect against operational downtime or the time needed to procure a spare device and to dispatch a technician to make the repair. To protect against operational downtime, redundancy of peripherals (N+1 redundancy) is introduced. With peripheral redundancy, if a peripheral malfunctions, the spare peripheral takes over operations of the malfunctioning peripheral without operator intervention. A repair technician can then be dispatched, somewhat less urgently, to restore redundancy to the system. Not only does peripheral redundancy enable the replacement of failed components with minimal downtime, it also allows for preventative maintenance.

Software Architecture

The System Release provides all the software components needed to build HA into telephony applications. The components include the hot-swap driver kit that is configured for specific chassis, management and fault detection software, and sample demonstration applications.

Figure 1 shows the architecture of the software components. The System Release provides all the components shown in the figure.

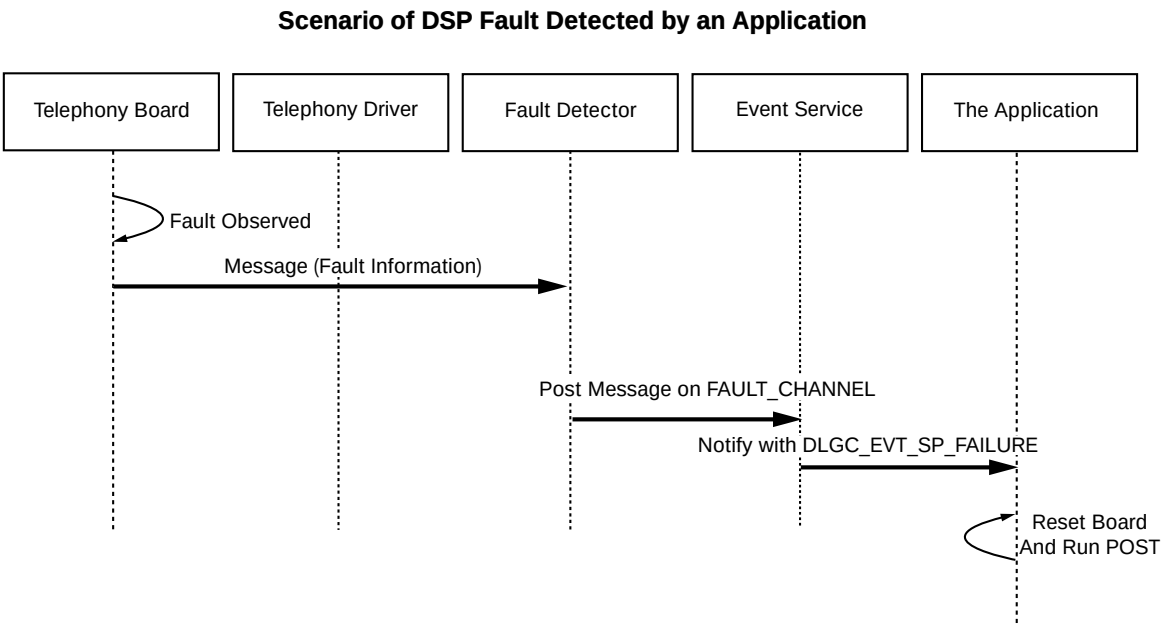


Figure 3. Example of fault detection

The architecture is best explained by the following scenarios. One shows how an application registers to receive notifications; a second shows the interactions when a fault occurs on a board.

Figure 2 shows the interactions between the components used to register for alarm and fault notifications from peripheral boards. Here is an outline of the application:

- The application creates an object of type DlgAdminConsumer by invoking the constructor of the class and passing in the channel type (e.g. FAULT_CHANNEL).
- This object then creates the necessary connections and sets up the communication between the various software components involved in the transaction.
- The application specifies filters for events it is interested in monitoring.
- A callback function HandleEvent() is implemented by the application invoked by the event notification framework whenever a specified event is observed.
- When an event is observed, the application receives all necessary information (i.e., the Channel name (FAULT_CHANNEL), the AUID of the board) to perform any actions.

Figure 3 shows an example of an application receiving an event when a DSP fault occurs on a peripheral board.

The main components in this transaction are the application, the device driver for the board, and management software that includes the Fault Detector and the Event Service along with the event notification framework. When a DSP on the telephony board fails, the firmware running on the board notifies the controller application by sending a message. This controller application is the Fault Detector that has registered for various alarms and faults at the time of initialization. When it receives this fault notification, it queues an event to the Event Service's event notification framework. Eventually, the host application is notified of the event on the registered callback function. At that time, the application can act upon the event and perform necessary operations or actions

Device Driver Interaction

The hot-swap system software resides between the operating system and hardware and acts as a hot-plug system monitoring software. The main tasks of this monitoring software are to detect hot-swap events, identify the board's required memory/interrupt

resources, and dynamically allocate them (or deallocate them upon board removal). To detect live insertion/removal of CompactPCI devices from the bus, the hot-swap engine can use one of the following methods:

- Polling the CompactPCI bus
- Polling the enumeration interrupt (ENUM#)

When the hot-swap system software detects the ENUM signal, it informs the Windows 2000 subsystem, specifically the Plug-n-Play Manager, of the detected event. The hot-swap system software has a well-defined, published interface with the Windows 2000 operating system (which supports plug/play events). Through this mechanism, the operating system is made aware of the newly-inserted device. The Plug-n-Play Manager provides a mechanism for device drivers and other applications to be notified when certain events occur on a specific device or on the system in general. These events include arrival and departure of device interfaces of the specified class, and device removal requests. When an event occurs, the Plug-n-Play Manager module calls the device driver's "add" or "init" entry points to process the initialization of the device drivers after allocating resources (e.g., interrupt, memory) as needed for the device.

Similarly, when the device is removed, the Plug-n-Play Manager calls the registered "remove" entry points in the device driver to handle the device removal requests and handle the freeing of allocated resources.

RSS Software

The RSS software is a separate package that can be installed before or after the System Release software. Refer to the release guide for a list of chassis tested with the release and for other system requirements.

For information about installing RSS software, see *Redundant System Slot Software for the ZT5550 High Availability Processor Board Software Manual* (RSS_Software_Manual.pdf). This manual and the executable file that installs the RSS software (ZRSS.exe) are located in the rss directory on the System Release CD-ROM. A sample application is included in this package that allows you to simulate a takeover or a failover of CPU cards. This sample application also demonstrates the use of the RSS API that is part of the SDK. The API allows you to program for:

- Fault configuration
- Isolation strategies
- Application notification
- Remote diagnostics

The RSS software provided by the System Release supports the Performance Technologies* ZT5084 CompactPCI system and the ZT5550 system master board (SBC). As mentioned earlier, this paper discusses the features of the CompactPCI system and SBC.

RSS High Availability (HA) API

The RSS HA API is discussed in the software manual (RSS_Software_Manual.pdf) provided by Performance Technologies. Some of the API functions are discussed in this section. For a telephony application to support RSS, it needs to register itself for notifications from the HA drivers and software components of the processor board. The rssmanager sample application included in System Release for Windows 2000 shows you how to use the APIs provided.

A host application uses the API by including the CompactHA.h header file and linking to the CompactHA.lib library file. Necessary parameter constants and types are defined in CompactHACnst.h and CompactHATypes.h. These header files and libraries are installed when the RSS software is installed on your system.

Table 1 lists some of the commonly used APIs

API	Description
HAConnect	Connects the host application to the HA framework.
HADisconnect	Terminates the connection with the host application.
HAConfigurationMode	Sets the current host's configuration mode. A host can only be placed in the configuration mode if it is not the active one.
HAEnableNotification	Enables the interrupt service routine to field the specified interrupt type. The callback function specified by the cbFunc parameter allows the host application to perform specific tasks based on the interrupt occurring. The application may register to receive notifications about faults and host state change by specifying different callback functions.
HADisableNotification	Prevents the host-application-defined interrupt service routine from fielding the specified interrupt types.
HAGetHostStatus	Reports the current host system status. Statuses available for query include system status and configuration information.
HAGetSlotID	Retrieves the physical slot information for the calling host.

Table 1: RSS API functions

PHS Software

The System Release software includes a hot-swap driver kit that can be configured for various chasses.

The telephony device drivers are configured automatically at install time for the specified chassis to perform the necessary hot-swap operations.

Hot-Swap Kit

The Hot-Swap Kit (HSK) is a CompactPCI hot-swap infrastructure product that supports automatic software connection and disconnection when boards are hot-inserted or hot-extracted. HSK provides the functional device drivers that fully support the native Device Driver Model for Windows 2000. HSK was the first product for Windows 2000 to achieve General Use Full Hot-Swap compliance as defined by PICMG 2.1, the CompactPCI hot-swap specification.

With the HSK installed on your CompactPCI system, you can:

- Insert and extract CompactPCI peripheral boards into and from a chassis with automatic software connection/disconnection of those boards and no rebooting.
- Use native application notification mechanisms to enable application monitoring of board insertions and removal requests. These notifications are delivered to the application via the Event Service API.

In conjunction with the HSK, the System Release device drivers automatically configure the PCI-to-PCI bridge windows for CompactPCI bus segments to ensure sufficient address space for hot insertions (since the BIOS-allocated windows are usually not sufficient). This is performed at driver initialization time after the telephony boards have been detected. Also, DCM, the configuration management GUI, provides physical slot geography of your system and displays the physical slot numbers, enabling operators to manage the systems better.

Management Software

The System Release provides management software that allows you to configure and monitor peripheral telephony devices. Fault detection, repair, and isolation components are also included.

The main components required to implement HA are:

- Fault management
- Alarm management
- Clock management
- Peripheral resource management

The System Release includes the Event Service API and Event Notification Framework. The API is used to register your application with the Event Notification Framework. The Framework is the subsystem for all operations, administration, maintenance, and provisioning (OAM&P) services to send asynchronous messages to registered telephony applications. For detailed information on the Event Service API and the Event Notification Framework, refer to the programming guides included in the release documentation. The framework contains various channels used to report a set of events that correspond to solicited or unsolicited actions of an operator.

Another library, the NCM API, provides an API to manage and monitor operations on the peripheral telephony devices. This API allows you to retrieve board-level information such as the physical slot ID, PCI Bus information, and CT Bus information. It also has functions to start/stop/quiesce boards.

Tables 2, 3, and 4 list the functions that would be used to develop an application to support PHS and redundant system slot features of the System Release. The sample applications included in the System Release — rgademo, rssmanager and pfmanager — illustrate the use of these APIs.

API	Description
NCM_IsHotSwapSystem	Determine if the system has hot-swap capabilities.
NCM_GetHotSwapBoardCount	Get the number of peripheral boards that currently are in the hot-swap capable system.
NCM_GetValueEx	Retrieve the value for a NCM database parameter.
NCM_SetValueEx	Set a value to a NCM database parameter.
NCM_DeallocValue	Release the memory allocated for the NCM database parameter.
NCM_GetFamilyDeviceByAUID	Retrieve the family type given the AUID of the board.
NCM_GetInstalledFamilies	Retrieve the family types of all installed boards.
NCM_GetInstalledDevices	Retrieve the list of installed boards.
NCM_StartDlgSrv	Start the Intel® Dialogic® system service.
NCM_GetDlgSrvState	Retrieve the state of the service.
NCM_StopDlgSrv	Stop the Intel Dialogic system service.
NCM_StartBoard	Start a single board.
NCM_StopBoard	Stop a single board.
NCM_RemoveBoard	Remove a board from the NCM database.
NCM_GetDialogicDir	Get a pointer to the System Release install folder.

Table 2: NCM API functions

API	Description
SRLGetAllPhysicalBoards	Get a list of boards currently installed in the system.
SRLGetVirtualBoardsOnPhysicalBoard	Retrieve the number of virtual boards on a physical board.
SRLGetSubDevicesOnVirtualBoard	Retrieve the number of sub-devices on a virtual board.

Table 3: SRL API functions

API	Description
DlgAdminConsumer::DlgAdminConsumer( )	Allows you to instantiate a consumer object. Each DlgAdminConsumer object must be associated with one event notification channel.
DlgAdminConsumer::DisableFilters( )	Disables a DlgAdminConsumer object's array of filters.
DlgAdminConsumer::EnableFilters( )	Enables a DlgAdminConsumer object's array of filters.
DlgAdminConsumer::getChannelName( )	Returns the channel name that a DlgAdminConsumer object monitors for incoming events.
DlgAdminConsumer::getConsumerName( )	Returns the name of the DlgAdminConsumer object. This name was associated with the consumer object at time of instantiation.
DlgAdminConsumer:: StartListening( )	Allows the DlgAdminConsumer object to begin monitoring its associated event notification channel for incoming events.
CEventHandlerAdaptor::HandleEvent( )	A virtual callback function invoked by the framework when an event is detected.

Table 4: Event Services API functions

Fault Management

Any malfunction of a hardware device is termed as a fault. The Event Service has components that are constantly monitoring the hardware devices by either polling or looking for a heartbeat signal. When the components detect a loss of the heartbeat signal, a fault is generated. Applications registered for the fault are notified appropriately.

There are two kinds of faults on any Intel® NetStructure™ board: Control Processor faults and Signal Processor faults. These faults are detected by the device driver via a mechanism setup between the device drivers and the individual kernel on each board (firmware). The device driver notifies the registered OAM&P services via function callbacks, which in-turn report the event on the FAULT_CHANNEL. The specific events that are reported are:

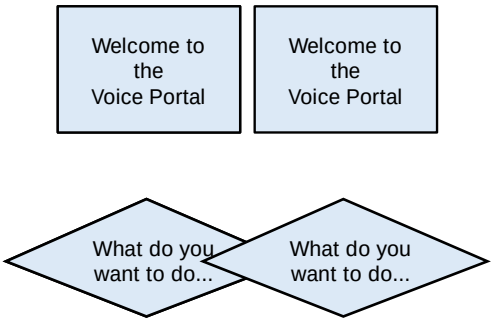
- **DLGC_EVT_CP_FAILURE** — Generated when a control processor failure occurs on an Intel® NetStructure™ board.
- **DLGC_EVT_SP_FAILURE** — Generated when a signal processor failure occurs on an Intel NetStructure board.

Detection and Recovery

Table 5 lists relevant events and recovery mechanisms.

Type	Why Fault is Generated	Actions Application Should Take
DLGC_EVT_CP_FAILURE	When the control processor that runs the firmware on a board fails or asserts for any reason.	Close all devices that are open on that physical board. Restart POST on the board and try to recover.
DLGC_EVT_SP_FAILURE	If a physical board has voice media capabilities, then there are specialized DSPs that could fail due to a variety of reasons. If such a failure occurs, then this event is reported to registered applications.	Close all devices that are open on that physical board. Restart the board and try to recover the channels on that board.

Table 5: Types of faults



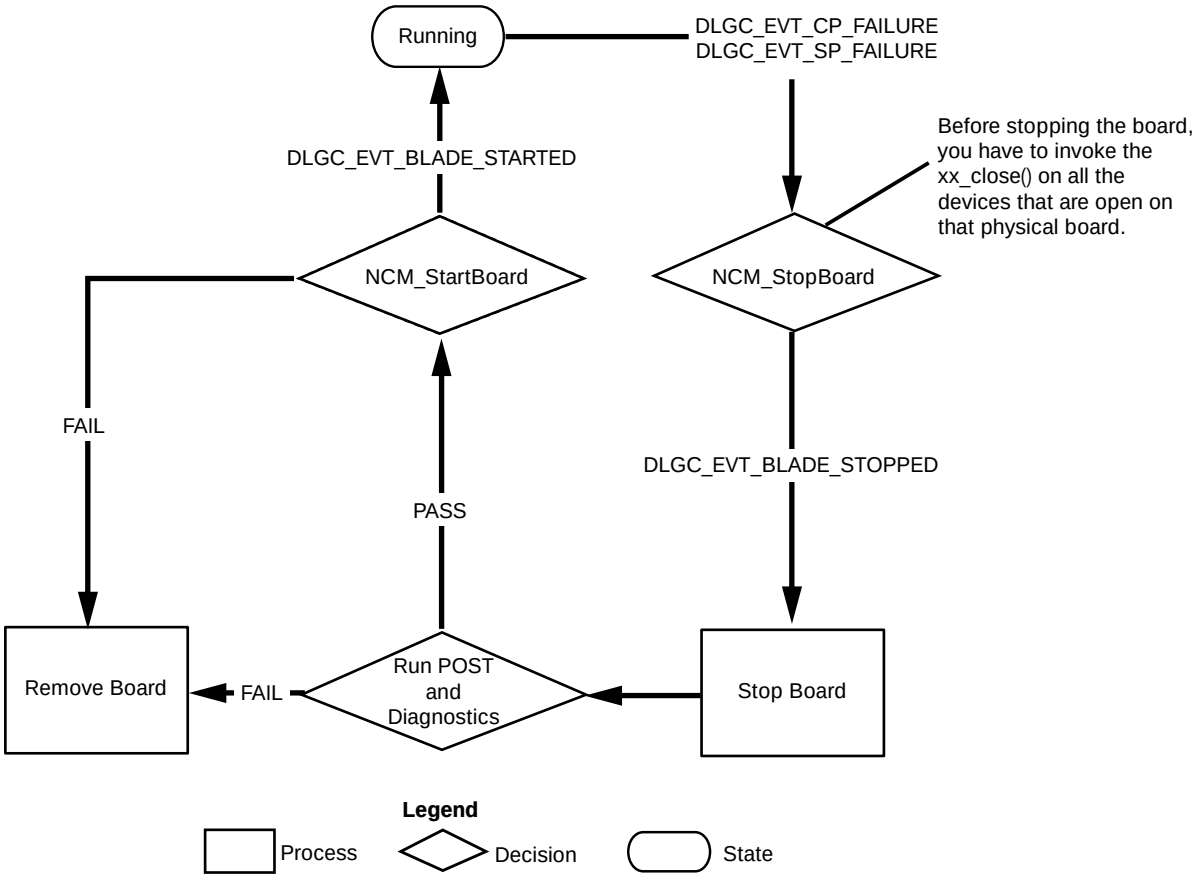


Figure 4. Fault detection and recovery

The flowchart in Figure 4 shows appropriate actions that are performed when faults occur on a board.

Alarm Management

Alarms are disruptions that occur on both the circuit and packet networks. On a circuit network, an alarm could be the T-1/E-1 cable being disconnected, a loss of frame signal, etc. On the packet network, an alarm could be the Ethernet cable being disconnected or a loss of signal with a router, etc. The Event Service, in collaboration with the Fault Detection Services, detects such alarms. The alarms on the circuit network are reported to the application via the NETWORK_ALARM_CHANNEL. The alarms on the packet (IP) network are reported to the application via the ENET_ALARM_CHANNEL.

Most of these alarms are also reported to a call control application via the Global Call alarm notification services, if the application has enabled such services. Appropriate Global Call events are generated into the SRL event queue and the application is notified. For example, when there is a RED alarm on the circuit connected to a T-1/E-1 span, a GCEV_BLOCKED event is generated to the application. When the alarm is cleared, a GCEV_UNBLOCKED event is generated.

Clock Management

The CT Bus can be programmatically configured for various setups. An OAM&P service, CT Bus Broker, monitors all activity of the CT Bus. When any failures occur the CT Bus broker, using the facilities of the Event Service, reports the events to registered applications on the CLOCK_EVENT_CHANNEL.

Types of alarms on the CT Bus include:

- **DLGC_EVT_CT_A_LINESBAD** — Occurs if the signal on the CT Bus Line A fails.
- **DLGC_EVT_CT_B_LINESBAD** — Occurs if the signal on the CT Bus Line B fails.
- **DLGC_EVT_LOSS_MASTER_SOURCE_INVALID** — Signals that the source used by the primary master board to drive the primary line has failed. The primary master board can use its own internal oscillator or a CT Bus Network Reference line as its clock source.
- **DLGC_EVT_NETREF1_LINEBAD** — Indicates that the signal on the CT Bus NetRef 1 line has failed.
- **DLGC_EVT_NETREF2_LINEBAD** — Indicates that the signal on the CT Bus NetRef 2 line has failed.

Almost all of these events simply provide information to the application. The application does not need to take any action when one of these events is observed, since the OAM&P services handle the clock fallback and failover mechanisms when something goes wrong.

Resource Management

Resource management for the Intel® NetStructure™ board is handled by the Standard Runtime Library (SRL) API and the Event Service API using the Event Notification Framework. Device enumeration and discovery are done using the SRL and NCM APIs.

- **SRLGetVirtualBoardsOnPhysicalBoard()** — Used to retrieve the number of virtual boards on a physical board identified by the AUID.
- **SRLGetSubDevicesOnVirtualBoard()** — Used to retrieve the number of sub-devices on a virtual board.

For example, suppose we have an Intel® NetStructure™ DMN160TEC board in the system. This physical PSTN network board has 16 T-1 or E-1 trunks, each represented by the device name dtiBn, where n represents a number from 1 to 16. By invoking the SRL function, SRLGetVirtualBoardsOnPhysicalBoard(), we would get 16 and the device type would be DTI. Using this information, we would build 16 device names (i.e., dtiB1, dtiB2, and so on, through dtiB16). Also, by invoking the SRL function, SRLGetSubDevicesOnVirtualBoard(), we would learn how many timeslots exist on each virtual board. If the DMN160TEC board were configured for a T-1 ISDN protocol, we would get 23 timeslots. If it were configured for an E-1 ISDN protocol, we would get 30 as the output from the function.

The hardware device detection is done using the event notification framework and the Event Service API. When a device is inserted on removed from the system, the Plug-n-Play Observer reports events to the Event Service that would deliver the same on the ADMIN_CHANNEL to registered applications.

The various events that are reported include:

- **DLGC_EVT_BLADE_ABOUT_TO_REMOVE** — Generated when the Device > Remove/Uninstall Device option is selected in the Intel® Dialogic® Configuration Manager (DCM).
- **DLGC_EVT_BLADE_ABOUTTOSTART** — Occurs when an individual board start command has been issued (either through the DCM's Device > Start Device option or programmatically with the NCM_StartBoard( ) function).
- **DLGC_EVT_BLADE_ABOUTTOSTOP** — Occurs when an individual board stop command has been issued (either through the DCM Device > Stop Device option or programmatically with the NCM_StopBoard( ) function).
- **DLGC_EVT_BLADE_DETECTED** — Indicates that a newly-inserted board has been detected by the System Release software and its initial configuration information has been stored in the NCM database.
- **DLGC_EVT_BLADE_REMOVED** — Generated when a board has been removed from the system and its configuration information has been deleted from the NCM database.
- **DLGC_EVT_BLADE_START_FAILED** — Occurs if an individual board's start sequence has failed. (The board start sequence can be initiated through DCM's Device > Start Device option or programmatically with the NCM_StartBoard( ) function).
- **DLGC_EVT_BLADE_STARTED** — Generated immediately after an individual board has been successfully started. (The board start can be initiated through DCM's Device > Start Device option or programmatically with the NCM_StartBoard( ) function).
- **DLGC_EVT_BLADE_STOPPED** — Generated immediately after an individual board has been successfully stopped. (The board stop can be initiated through DCM's Device > Stop Device option or programmatically with the NCM_StopBoard( ) function).
- **DLGC_EVT_SYSTEM_ABOUTTOSTART** — Occurs when a system start command has been issued (either through the DCM's System > Start System option or programmatically with the NCM_StartDlgSrv() function).
- **DLGC_EVT_SYSTEM_ABOUTTOSTOP** — Occurs when a system stop command has been issued (either through the DCM's System > Stop System option or programmatically with the NCM_StopDlgSrv( ) function).
- **DLGC_EVT_SYSTEM_STARTED** — Generated immediately after the system has been successfully started. (The system start can be initiated through DCM's System > Start System option or programmatically with the NCM_StartDlgSrv( ) function).
- **DLGC_EVT_SYSTEM_STOPPED**— Generated immediately after the system has been successfully stopped. (The system stop can be initiated through DCM's System > Stop System option or programmatically with the NCM_StopDlgSrv( ) function).

The state chart in Figure 5 illustrates the various events listed above and shows a state machine implemented in the PFMDemo application included in the System Release.

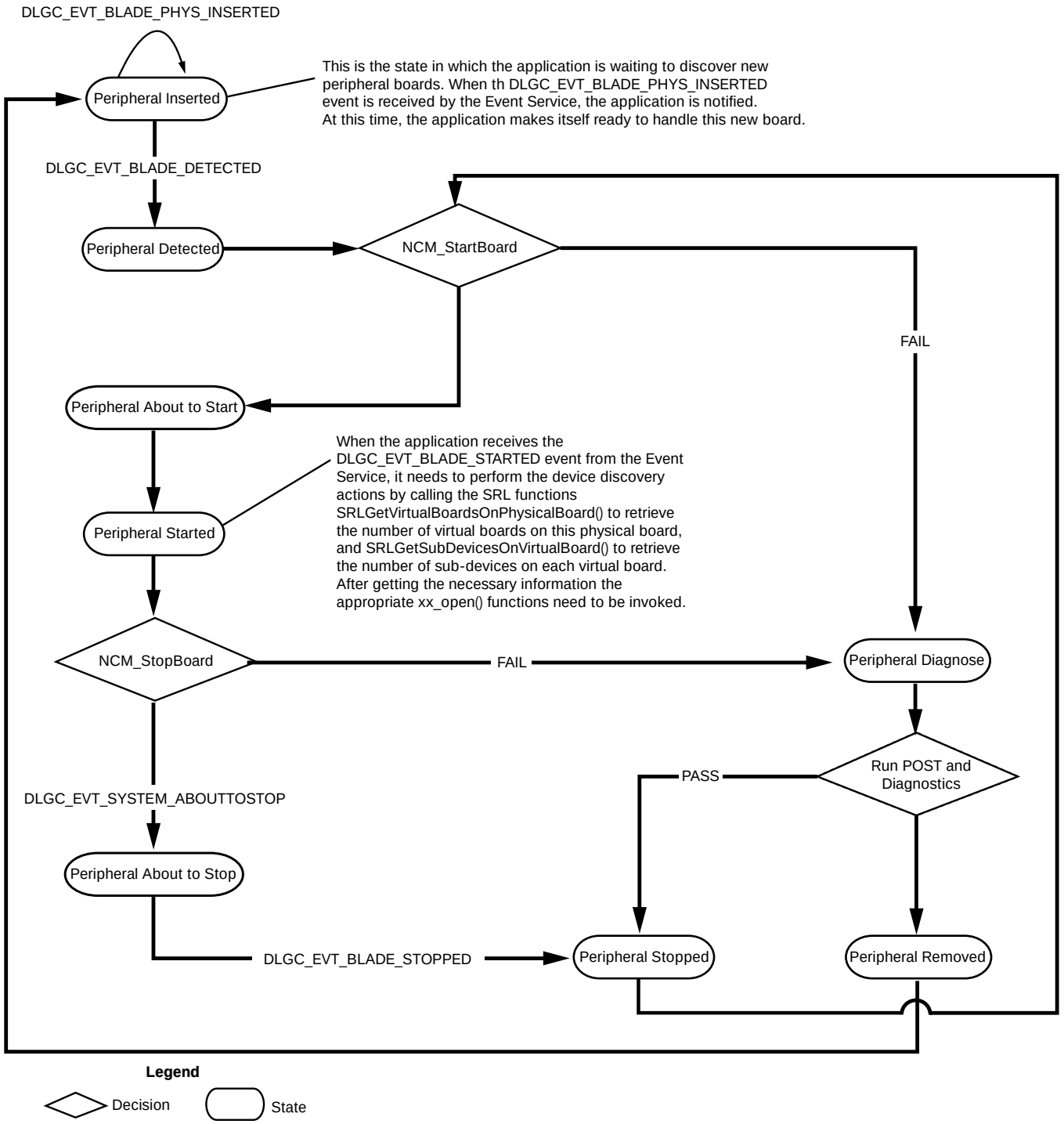


Figure 5: Event services state machine

Executing POST on Boards

When a peripheral board is detected in the system, it is recommended that you execute POST on that device to ensure that the hardware is good and functional. The System Release provides POST utilities that can be executed individually based on the type of hardware and the family to which it belongs. Figure 6 illustrates how this can be done. This code snippet is part of the pfmanager sample application of the System Release.

The Diagnose() function shows you how to invoke the DM3 and IPT post utilities.

```
NCMRetCode
CDevice::Diagnose ()
{
 long lBoardId = 0;
 char szCmd[256];
 char szDir[128];
 DWORD dwRc;
 NCMRetCode ncmRet;
 int iLen = 128;

 ncmRet = NCM_GetDialogicDir (“DNASDKDIR”, &iLen, szDir);
 if (ncmRet == NCM_SUCCESS)
 {
 if (!strcmp(m_Family.name,”DM3”))
 {
  //run dm3 post silently
  sprintf (szCmd,”%s\\Dm3Post -b%d -s%d”, szDir,m_PCIBus,
m_PCISlot);
  dwRc = RunProgram (szCmd);
  switch (dwRc)
  {
  case 0:
  case DM38BITOK:
  case DM316BITOK:
  case DTI168BITOK:
  case DTI1616BITOK:
   ncmRet = NCM_SUCCESS;
   break;

  default:
   ncmRet = NCME_GENERAL;
  }
 }
 else if (!strcmp(m_Family.name, “IPT”))
 {
  //run ipt post.
    .
    .
    .
 }
 else
 {
  ncmRet = NCME_GENERAL;
 }
 }

 return ncmRet;
}
```

Figure 6. Executing POST

The RunProgram() function in Figure 7, shows a way of creating a Windows process, then waiting for that process to complete execution.

```
DWORD
CDevice::RunProgram (char *lpCommandLine)
{
 STARTUPINFO si;
 PROCESS_INFORMATION pi;
 DWORD retcode;

 si.cb = sizeof (STARTUPINFO);
 si.lpReserved = NULL;
 si.lpDesktop = NULL;
 si.lpTitle = NULL;
 si.dwFlags = 0;
 si.wShowWindow = SW_SHOW; //hide the program
 si.cbReserved2 = 0;
 si.lpReserved2 = NULL;

 pi.hProcess = NULL;
 pi.hThread = NULL;

 if (!CreateProcess (NULL, lpCommandLine, NULL, NULL, false,
      CREATE_NEW_PROCESS_GROUP, NULL, NULL, &si, &pi))
 {
 return -1;
 }

 switch (WaitForSingleObject (pi.hProcess, 120000))
 {
 case WAIT_ABANDONED:
 case WAIT_TIMEOUT:
  return -1;
  break;

 default:
  GetExitCodeProcess (pi.hProcess, &retcode);
  break;
 }

 if (pi.hProcess)
 CloseHandle (pi.hProcess);

 if (pi.hThread)
 CloseHandle (pi.hThread);

 return retcode;
}
```

Figure 7. Executing POST (continued)

CompactPCI Platforms

This section provides some details on the CompactPCI systems that have been used to validate the System Release software.

Note that the ZT5084 platform is fully supported by the System Release software. However, there is limited support for the ZT5085 platform in the form of PHS. RSS is not supported. Some specific configuration steps need to be performed when setting up the System Release software in the ZT5085 platform.

Intel® NetStructure™ ZT5084 10U High Availability Platform

This HA CompactPCI platform provides a carrier-grade computing system for demanding, mission-critical applications. The ZT5084 platform supports five-nines (99.999%) availability with built-in redundancy for active system components including system-slot CPU boards, power supplies, and alarming. These components are hot-swappable to simplify replacement and minimize service time.

The ZT5084 platform is ideal for telecom applications requiring high system availability (e.g., enhanced services, media gateways, broadband access servers, or any critical computing server platform destined for the central office). The hardware-based failover and simplified HA driver model shorten development time for telecom equipment designers, while redundant system-slot architecture enables more efficient use of expensive I/O resources. This CompactPCI system has 12 slots available for peripheral devices.

The ZT5550 High Availability Processor Board is the only supported processor board for the ZT5084 platform. This 6U, CompactPCI board is designed for redundant processing configurations where very high levels of system availability are required. Its architecture is tailored for demanding applications such as those servicing the telecom network and the Internet.

The ZT5550 High Availability Processor Board supports up to 12 CompactPCI peripheral boards and, when used with a second ZT5550 board, can provide five-nines availability. It features the Intel, Pentium® III processor low-power module running at 500MHz, is hot-swappable, and includes several on-board peripherals and optional I/O expansion features. The board occupies one or two slots, depending on the configuration.

Intel® NetStructure™ ZT5085 12U Redundant Host Packet Switched Platform

The Intel® NetStructure™ ZT5085 12U Redundant Host Packet Switched Platform features a PICMG* 2.16-compatible mid-plane supporting redundant-host architecture for I/O-intensive applications. It is one of several modular telecom building blocks from Intel, providing OEM equipment designers with a carrier-grade, standards based, HA computing platform for demanding mission-critical applications.

The Platform supports five-nines availability with built-in redundancy for active system components including Ethernet switches, chassis management modules, power supplies, and fan trays. Redundant chassis management modules make it possible to manage multiple SBCs and conduct chassis diagnostics remotely for enhanced system reliability. Ethernet signals are routed across the mid-plane without the use of cables, saving time in set-up, maintenance, and repair; and minimizing the thermal challenges of traditional cabling methods.

The Platform is designed to interoperate with all members of the Intel® NetStructure™ family of packet-switched products and with third-party boards meeting the PICMG 2.16 specifications.

There are two supported processor boards for the ZT5085 Platform, the Intel® NetStructure™ ZT5504 and Intel® NetStructure™ ZT5524 boards.

The ZT5504 processor board is a 2.16-compliant processor board that offers excellent value with an optimized feature set to support a broad range of telecom and Internet applications. Modular and standards-based, the ZT5504 supports efficient time-to-market

development. Completely validated with the Intel® NetStructure™ family of packet switched backplane (PSB) products, it is designed to be interoperable with third-party 2.16-compliant components. The board features a 1GHz, low-power Pentium III processor with 512 Mbytes to 1 Gbyte ECC SDRAM.

The ZT5524 high-performance processor board is a standards-based building block designed for carrier-grade telecom and Internet applications requiring exceptional processing power and HA. The dual processor/redundant host board is PICMG* 2.16-compliant and offers configurable HA, I/O expansion, and 66MHz CompactPCI bridging features. A generous set of on-board embedded features and reliable, off-the-shelf architecture address the integration and reliability requirements of OEM systems builders. This board features a single or dual 933MHz Pentium III processor that supports symmetric multiprocessing in single CompactPCI slot with a single 168-pin, right-angle DIMM module socket. It supports up to 1GB PC133 SDRAM memory.

Adding PHS support

Certain changes are required to support PHS in an existing telephony application. The changes can be summarized into the following steps:

1. Enumerate the peripheral device events to which you want the application to listen. For example, you may only want events on the FAULT_CHANNEL and the ADMIN_CHANNEL.
2. Register the application to receive the peripheral device events via the Event Service. This is done using the Event Service APIs.
3. Build a state machine in your application (similar to the one shown in Figure 5) to properly handle the various events.

Adding Redundant System Slot Support

An existing telephony application needs to make certain changes to support the RSS feature. These can be summarized as follows:

1. Identify the chassis vendor's device driver events. This should be available from the vendor's documentation. The RSS Manager sample application provided with the System Release illustrates the use of the ZT5084 chassis and the ZT5550 SBC boards. It registers for specific events from the device driver provided by the chassis vendor.
2. Register the application to receive the events on the ADMIN_CHANNEL of the Event Service. This will enable you to monitor the activities of the Intel, NetStructure' boards.
3. Register the application to receive events from the RSS HA framework provided by the chassis vendor. This will enable the application to monitor the activities of the processor boards in the system.
4. Build a state machine into your application that will perform appropriate actions when the Event Service reports the board level events.

Sample Applications

The System Release software contains sample applications that demonstrate the use of the HA features supported. There are four applications:

1. **RSS Manager** — Demonstrates the use of the RSS HA APIs provided by the ZT5550 system master board from Performance Technologies. The application monitors the activities on the SBC. When an active SBC goes down (due to a friendly takeover, hostile takeover, or critical fault), the application is notified via a callback function that indicates that a takeover occurred and that the standby SBC is now the current active host. At this time, the application performs necessary peripheral operations on the Intel NetStructure boards.

2. **Peripheral Fault Manager (PFM)** — Allows users to start/stop peripheral boards installed in a system. It demonstrates the use of Event Notification Framework events retrieved by using the Event Service API.
3. **Revenue Generating Application (rgademo)** — This call control application works with the Peripheral Fault Manager application and the RSS Manager application. It also registers itself as a consumer to the various events generated by the Event Notification Framework. It monitors activities on all peripheral devices installed in the system. When a telephony peripheral is inserted into the system, appropriate events are detected by this application. After the peripheral is initialized and started, ISDN calls are made in a loopback fashion for demonstration purposes. Refer to the user's guide for more details on how to execute the application and set up the boards.
4. **HA Demo** — This demonstration application, provided by Performance Technologies, illustrates the use of the RSS HA APIs and the framework provided by the ZT5550 system master board in the ZT5084 platform.

The System Release contains a demo guide, "High Availability for Windows Demo Guide," that describes these sample applications in detail.

For More Information

Learn more about Intel Dialogic System Release 6.0 CompactPCI for Windows 2000 and download the software at http://www.intel.com/network/csp/products/8326web.htm.

Appendix: Glossary

API	Application programming interface
AUID	Addressable unique identifier
DSP	Digital signal processor
HA	High availability
HSK	Hot-Swap Kit
NCM	Native configuration manager
OAM&P	Operations, administration, maintenance, and provisioning
PHS	Peripheral hot-swap
PICMG	PCI Industrial Manufacturing Group
RSS	Redundant system slot
SBC	Single-board computer

To learn more, visit our site on the World Wide Web at **http://www.intel.com**.

1515 Route Ten
Parsippany, NJ 07054
Phone: 1-973-993-3000

INFORMATION IN THIS DOCUMENT IS PROVIDED IN CONNECTION WITH INTEL® PRODUCTS. NO LICENSE, EXPRESS OR IMPLIED, BY ESTOPPEL OR OTHERWISE, TO ANY INTELLECTUAL PROPERTY RIGHTS IS GRANTED BY THIS DOCUMENT. EXCEPT AS PROVIDED IN INTEL'S TERMS AND CONDITIONS OF SALE FOR SUCH PRODUCTS, INTEL ASSUMES NO LIABILITY WHATSOEVER, AND INTEL DISCLAIMS ANY EXPRESS OR IMPLIED WARRANTY, RELATING TO SALE AND/OR USE OF INTEL PRODUCTS INCLUDING LIABILITY OR WARRANTIES RELATING TO FITNESS FOR A PARTICULAR PURPOSE, MERCHANTABILITY, OR INFRINGEMENT OF ANY PATENT, COPYRIGHT OR OTHER INTELLECTUAL PROPERTY RIGHT.

Intel products are not intended for use in medical, life saving, life sustaining applications.

Intel may make changes to specifications and product descriptions at any time, without notice.

Intel, Intel Dialogic, Intel NetStructure, Pentium, and the Intel logo are trademarks or registered trademarks of Intel Corporation or its subsidiaries in the United States and other countries.

*Other names and brands may be claimed as the property of others.

Performance tests and ratings are measured using specific computer systems and/or components and reflect the approximate performance of Intel products as measured by those tests. Any difference in system hardware or software design or configuration may affect actual performance. Buyers should consult other sources of information to evaluate the performance of systems or components they are considering purchasing. For more information on performance tests and on the performance of Intel products, reference http://www.intel.com/procs/perf/limits.htm or call (U.S.) 1-800-628-8686 or 1-916-356-3104.

Printed in the USA Copyright © 2003 Intel Corporation All rights reserved. Printed on recycled paper. 03/03 00-8549-001