Intel® Dialogic® System Release 6.1 for Linux

Release Update

January 22, 2007

05-2466-016

Publication Date: January 22, 2007

Document Number: 05-2466-016

Intel
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

About This Publication

This section contains information about the following topics:

Purpose

This Release Update addresses issues associated with Intel® Dialogic® System Release 6.1 for Linux. In addition to summarizing issues that were known as of the release's general availability, this Release Update will continue to be updated to serve as the primary mechanism for communicating new issues that arise after the release date.

Intended Audience

This Release Update is intended for all users of System Release 6.1 for Linux.

How to Use This Publication

This Release Update is organized into four sections (click the section name to jump to the corresponding section):

- Document Revision History: This section summarizes the ongoing changes and additions that are made to this Release Update after its original release. This section is organized by document revision and document section.
- Post-Release Developments: This section describes significant changes to the system release subsequent to the general availability release date. For example, the new features provided in Service updates are described here.
- Release Issues: This section lists issues that may affect the system release hardware and software. The primary list is sorted by issue type, but alternate sorts by PTR number, by product or component, and by Service Update number are also provided.
- Documentation Updates: This section contains corrections and other changes that apply to the System Release documentation set that could not be made to the documents prior to the release. The updates are organized by documentation category and by individual document.

Related Information

Refer to the following for additional information:

- For further information on issues that have an associated PTR number, you may use the web-enabled PTR Lookup tool at:
  http://membersresource.intel.com/search/ptrs/ptrsearch.asp
  When you select this link, you will be asked to either LOGIN or JOIN. If you have any questions about this new tool or wish to provide feedback regarding this tool, please use the contact page on the Telecom Support Services website at:
  http://resource.intel.com/telecom/support/webcontact.htm
- For Technical Support, visit the Telecom Support Resources website at the following link:
  http://developer.intel.com/design/telecom/support/index.htm
- For Intel community development support, visit the Open System Release website at the following link:
  http://www.opensystemrelease.com
  By providing source code and development services, Intel encourages our customers to participate in the improvement and evolution of future system releases.
- For Products and Services Information, visit the Telecom Products website at the following link:
  http://www.intel.com/design/telecom/index.htm
- For Sales Offices and other contact information, visit the Buy Telecom Products website at the following link:
  http://www.intel.com/buy/networking/telecom.htm
- For information about the products and features supported in this release, see the *Intel Dialogic System Release 6.1 for Linux Release Guide* document that is included as part of the documentation set for this release.

Document Revision History

This Revision History summarizes the changes made in each published version of the Release Update for Intel® Dialogic® System Release 6.1 for Linux, which is a document that is periodically updated throughout the lifetime of the release.

Document Rev 16 - published January 22, 2007

Updated for Service Update 241.

In the Post-Release Developments section:

- Added New ANI/DNIS-Enabled Parsing Tool (ADEPT) for PBX Integration Boards.
- Updated the New Diagnostics Management Console section to add more tools that can now be executed: AppMon, Castrace, Isdntrace, Dlgsnapshot, Dm3post, Debugangel, and Pdktrace
- Updated the New Version of its_sysinfo Tool section to add new Linux Package Info that is now included in the *its_sysinfo.htm* file.

In the Release Issues section:

- Added the following resolved problems: IPY00031534, IPY00032794, IPY00032803, IPY00033185, IPY00034105, IPY00034495, IPY00034678, IPY00035506, IPY00036044, IPY00036247.

In the Documentation Updates section:

- Added a documentation update to the following document because of a new feature in the Service Update: Diagnostics Guide.
- Added IPY00006024 (PTR 29612) under PBX Integration Board User's Guide.
- Added the ADEPT for PBX Integration Boards User's Guide to the bookshelf. Also, added information pertaining to the Linux release.

Document Rev 15 - published January 2, 2007

Updated for Service Update 239.

In the Post-Release Developments section:

- Added Dynamically Changing the Transmit Time Slot on IP Media Devices.

In the Release Issues section:

- Added the following resolved problems: IPY00010760 (PTR 36647), IPY00011037 (PTR 36677), IPY00010929 (PTR 36497), IPY00028222 (PTR 36483), IPY00028460 (PTR 36298), IPY00033563, IPY00033912, IPY00034036, IPY00034406, IPY00034559, IPY00034627, IPY00034841, IPY00035822, IPY00035831.

In the Documentation Updates section:

- Added updates to *IP Media Library API for Linux and Windows Programming Guide* and *IP Media Library API Library Reference* for dynamically changing the transmit time slot on IP media devices.

Document Rev 14 - published December 11, 2006

Updated for Service Update 237.

In the Release Issues section:

- Added the following resolved problems: IPY00035860, IPY00036035.

Document Rev 13 - published November 17, 2006

Updated for Service Update 234.

In the Release Issues section:

- Added the following resolved problem: IPY00035660.
- Added the following known problems: IPY00010760 (PTR 36647), IPY00010929 (PTR 36497), IPY00011037 (PTR 36677), IPY00028222 (PTR 36483), IPY00028460 (PTR 36298). These were incorrectly listed as resolved in Service Update 232.

Document Rev 12 - published November 10, 2006

Updated for Service Update 232.

In the Post-Release Developments section:

- Added Runtime Control of Double Answer for R2MF.
- Added Support for D/4PCI Board.
- Added two more supported boards, Intel NetStructure® DM/V2400A and DMV4800BC Combined Media Boards, to Additional Voice Channels on Clear Channel Media Loads.
- Added Support for PCI Express Boards.
- Added updates to Additional Supported Operating System Distributions (OSDs) for Red Hat* Enterprise Linux Versions 3.0 and 4.0.

In the Release Issues section:

- Added the following resolved problems: IPY00010760 (PTR 36647), IPY00010929 (PTR 36497), IPY00011037 (PTR 36677), IPY00028222 (PTR 36483), IPY00028460 (PTR 36298), IPY00029922 (PTR 35353), IPY00033698, IPY00034606, IPY00034738.

Document Rev 11 - published September 5, 2006

Updated for Service Update 226.

Note: The Release Issues section has been modified to show issues by Change Control System defect number and by PTR number.

In the Post-Release Developments section:

- Added Support for 12 GB RAM.
- Added Support for Reporting Billing Type.
- Added Dynamically Adding and Deleting SS7 Circuit Groups.
- Added Global Call Support for Time Slots on SS7 Boards Running in DTI Mode.
- Added Additional Voice Channels on Clear Channel Media Loads.
- Added Media Channel Reset Capability (Stuck Channel Recovery).

In the Documentation Updates section:

- Added a revised statement on system requirements under Release Guide.

Document Rev 10 - published August 21, 2006

Updated for Service Update 225.

In the Post-Release Developments section:

- Added Additional Supported Operating System Distributions (OSDs).
- Added New Diagnostics Management Console.
- Added More Configurations for Optional Use of Sharing of Timeslot (SOT) Algorithm.
- Added Ability to Send and Receive DPNSS End to End Messages.
- Added Time Stamp for Tone-On/Off Events.
- Added OA&M Error Cleanup.

Document Rev 09 - published June 29, 2006

Updated for Service Update 217.

In the Post-Release Developments section:

- Added Additional Supported Operating System Distributions (OSDs).
- Added New Runtime Trace Facility (RTF) Manager.
- Added New Application Monitor.
- Added Improved Tracing and Error Reporting.

Document Rev 08 - published June 7, 2006

Updated for Service Update 213.

In the Post-Release Developments section:

- Added New Media Load for DMV600BTEC Boards.
- Added Removal of IP Gateway R4 and IPML Gateway Demos.

In the Release Issues section:

- Added the following resolved problems: 36598.

In the Documentation Updates section:

- Removed IP Gateway (Global Call) Demo Guide since the demo is no longer supported.

Document Rev 07 - published April 28, 2006

Updated for Service Update 208.

In the Post-Release Developments section:

- Added New Status Monitor (statusmon) Application.
- Added New Version of Runtime Trace Facility (RTF) Tool.
- Added New Version of Get Version (Getver) Tool.
- Added New Version of Intel Telecom Subsystem Summary (its_sysinfo) Tool.

In the Release Issues section:

- Added the following resolved problems: 35670, 36644, 36670, 36790.

In the Documentation Updates section:

- Removed documentation updates for the Diagnostics Guide since these updates have been incorporated into the latest version of the Diagnostics Guide which is now available online.

Document Rev 06 - published April 18, 2006

Updated for Service Update 205.

In the About This Publication section:

- Added related information for the Open System Release website.

In the Post-Release Developments section:

- Added Support for Open Source Drivers.
- Added Optional Use of Sharing of Timeslot (SOT) Algorithm.
- Added Deprecation of Dlgcsnmpd and Orbacus Init Scripts.
- Added Removal of Unused Header Files.

In the Release Issues section:

- Added the following resolved problems: 36656, 36657, 36682, 36725, 36727, 36817, 36833.
- Added known problem noting that MLFN media load for 60 channel fax is not supported on DMV600BTEC board.

Document Rev 05 - published April 5, 2006

Updated for Service Update 204.

In the Release Issues section:

- Added the following resolved problems: 33530, 36452, 36481, 36550.
- Added the following known permanent problems: 35891.
- Added the following known problems: 36581, 36877.

Document Rev 04 - published March 10, 2006

Updated for Service Update 198.

In the Post-Release Developments section:

- Added Additional Supported Operating System Distributions (OSDs).
- Added Dynamic Detection of ETSI FSK Protocols.
- Added **dx_stopch( )** EV_NOSTOP Mode Support for DM3 Boards.

In the Release Issues section:

- Added the following resolved problems: 31991, 34433, 35955, 36439, 36519, 36577, 36655.
- Added the following known problems: 35851, 36104, 36452, 36481, 36550, 36564, 36644, 36656, 36657, 36659, 36670, 36682, 36697, 36714, 36727, 36817, 36818.

In the Documentation Updates section:

- Added a new procedure under Software Installation Guide.
- Added PTR# 36726 under Global Call E1/T1 CAS/R2 Technology Guide.
- Added PTR# 35565 under MSI API Library Reference.
- Added PTR# 34546 and PTR# 35667 under Voice API Programming Guide.
- Added new procedures under Pigeon Point Systems Linux Hot Swap Kit User Guide.

Document Rev 03 - published December 13, 2005

Updated for Service Update 189.

In the Post-Release Developments section:

- Added Additional Supported Operating System Distributions (OSDs).

- Added Supported Kernel and GCC Versions.
- Added New Media Loads for DMV1200BTEC Boards.
- Added GSM-AMR-NB Coder Support for IPT Boards.
- Added Tcl/Tk No Longer Required for QScript Utilities Support.
- Added System Logging Integrated with Runtime Trace Facility (RTF).
- Added Support for Multiple SS7 Boards in the Same System.

In the Release Issues section:

- Added the following resolved problems: 35532, 35763, 35819, 35842, 35860, 35861, 35874, 35923, 35925, 36075, 36093, 36123, 36125, 36402, 36408.
- Added the following known permanent problems: 36102, 36155, 36198, 36381.
- Added the following known problems: 36294, 36322, 36394.

In the Documentation Updates section:

- Removed documentation updates for the Software Installation Guide since these updates have been incorporated into the latest version of the Software Installation Guide which is now available online.
- Added PTR# 36306 and PTR# 36343 under Administration Guide.
- Added correction that Tcl/Tk no longer needs to be installed under Administration Guide and Diagnostics Guide.
- Added correction that states support for multiple SS7 boards in the same system under Global Call SS7 Technology Guide.
- Added PTR# 35969 under ISDN Software Reference.

Document Rev 02 - published October 7, 2005

Updated for Service Update 171.

In the Post-Release Developments section:

- Added Service Update for System Release 6.1 for Linux.
- Added Support for IP Boards and Features.
- Added Support for Redundant Host (RH).
- Added Support for Peripheral Hot Swap (PHS) on Additional Compute Platforms.
- Added New Media Load for DMV4800BC Boards.
- Added Dlgsnapshot Tool's Autodump Feature Now Disabled by Default.

In the Release Issues section:

- Added the following resolved problems: 35554, 35615, 35794, 35821, 35853, 35888.
- Added the following known problems: 36048, 36075, 36093, 36102, 36104, 36123, 36125, 36131.

In the Documentation Updates section:

- Added PTR# 36105 under Release Guide.
- Added PTR# 35965 under Global Call API Library Reference.

Document Rev 01 - published August 31, 2005

Initial Version of document.

Post-Release Developments

This section describes significant changes to the system release subsequent to the general availability release date.

1.1 Service Update for System Release 6.1 for Linux

A Service Update for System Release 6.1 for Linux is now available. Service Updates provide fixes to known problems, and may also introduce new functionality. New versions of the Service Update will be released periodically. This Release Update will document the features in the Service Updates.

Depending on whether you already have a version of System Release 6.1 for Linux on your system, installing the Service Update will give you either a **full install** or an **update install**:

- If you don't have an existing version of System Release 6.1 for Linux on your system, installing the Service Update gives you a **full install** of the release. You can select the features that you want to install, for example, Intel NetStructure® DMV/DMN/DMT, Global Call Protocols, Documentation, etc.
- If you have an existing version of System Release 6.1 for Linux on your system, installing the Service Update gives you an **update install**. The update install gives you the latest software for the features that you selected when you did the full install of the system release that is currently on your system.

1.2 New ANI/DNIS-Enabled Parsing Tool (ADEPT) for PBX Integration Boards

With the Service Update, you can now use a tool called the ANI/DNIS-Enabled Parsing Tool (ADEPT), which is a set of software modules that functions as a portable, generic, Calling Party Identification (CPID), Automatic Number Identification (ANI), and Dialed Number Identification Service (DNIS) display parser. By using ADEPT, a user can very easily update display parsing rules that are specific to a site or an application.

Previous display parsers for the PBX Integration Boards could not be modified without rebuilding the firmware. With the ADEPT implementation, the CPID parsing is no longer controlled in the firmware. Instead, it is controlled in the d42 library with appropriate rules changes. The advantage of ADEPT is that a customer can change the display parser just by changing the rules file. The rules file is implemented as a simple text file. ADEPT provides significant productivity advantages by eliminating the need to rebuild firmware. No recompilation is needed in any subsystem.

Both the Intel® Dialogic® D/82JCT-U and D/42JCT-U boards support the ADEPT functionality.

Detailed information about this feature is provided in a new document, *ADEPT for PBX Integration Boards User Guide,* which has been added to the online bookshelf for System Release 6.1 for Linux.

Note: Additional information for Linux and Windows has been added to the Section 3.4.22, "ADEPT for PBX Integration Boards User's Guide", on page 128 in the Documentation Updates section of this document.

1.3 Dynamically Changing the Transmit Time Slot on IP Media Devices

With the Service Update, the ability to dynamically change the transmit time slot on IP Media (ipmBxCx) devices is provided. The feature is supported on Intel NetStructure® IPT boards only.

1.3.1 Feature Description

The H.110 TDM transmit time slots for IP Media devices are allocated by the time slot allocation during the board initialization time. In previous releases, this allocation could not be changed at runtime. Starting with this Service Update, the TDM transmit time slots on the IP Media devices associated with IPT boards can be dynamically changed later at runtime using the **ipm_SetParm( )** IP Media Library (IPML) function and a new PARMCH_TX_TIMESLOT parameter that has been created for IPM_PARM_INFO data structure eIP_PARM define to support this feature.

The companion **ipm_GetParm( )** function can be used to retrieve the transmit time slot to which the IP Media device is currently allocated. The newly assigned transmit time slot

allocation remains in effect until the device is closed or another request to change the transmit time slot allocation is issued.

Note: Upon a device close (that is, **ipm_close ( )**), the Tx time slot for the specified device will revert back to the static allocated value assigned during board initialization.

Caution*: Assigning transmit time slots incorrectly can cause physical damage on the IPT or any other H.110 board when two or more devices are transmitting on the same TDM time slot. It is the responsibility of the application to ensure that the transmit time slot of each IP Media device is allocated to a time slot that has no other device transmit time slots allocated to it.*

1.3.2 New PARMCH_TX_TIMESLOT Parameter

A new PARMCH_TX_TIMESLOT parameter has been created for IPM_PARM_INFO data structure eIP_PARM define to support this feature:

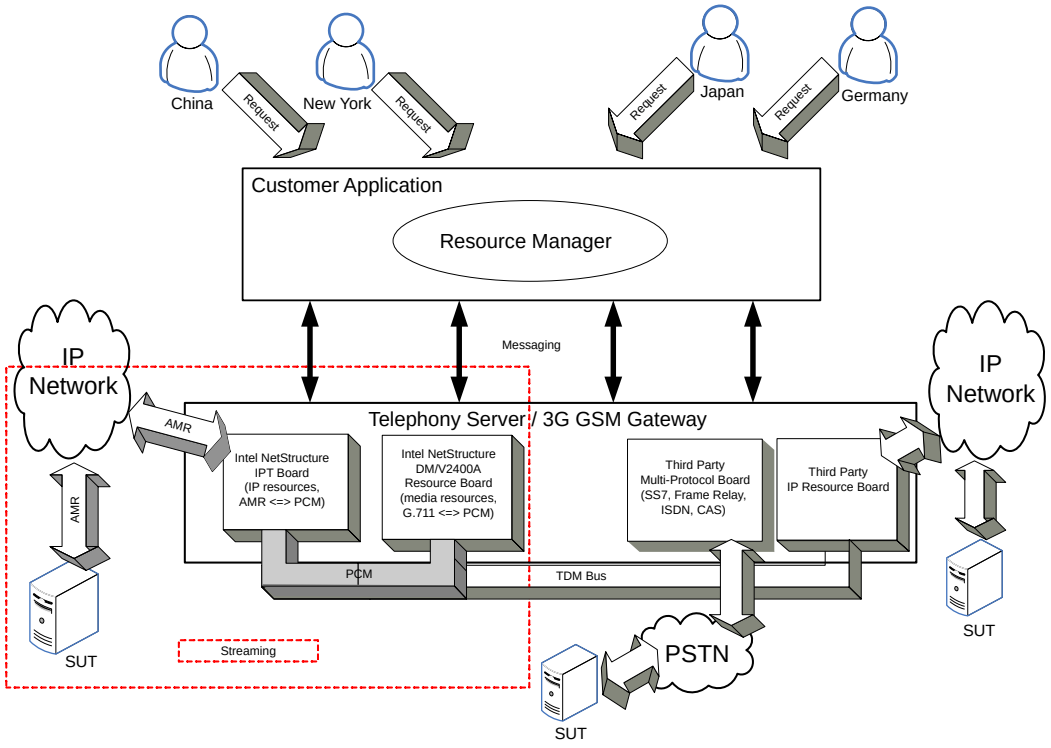
PARMCH_TX_TIMESLOT
Identifies a transmit time slot value for an IP Media device that can be dynamically set or retrieved at runtime.
Type: unsigned short. Valid values: 0 to 4095.

1.3.3 Use Cases

A number of use case scenarios that describe the usefulness of this feature are described in the following subsections. The system configuration for use cases 1 and 2 is described first.

System Configuration for Use Cases 1 and 2

The figure below shows the system configuration for use cases 1 and 2.



In this configuration, the customer application is designed to expect time slots in the range 1 to 240 to be assigned to the DM/V2400A media resources for transmitting data and time slots in the range 2001 to 2240 for receiving data (see the following table).

Boards	Static Tx TDM Time Slots	Tx TDM Time Slots	Rx TDM Time Slots	TDM Time Slot Pool
DM/V2400A	1 to 240	1 to 240	2001 to 2240	---
Third Party Boards	---	---	---	2241 to 4096

The application has an administration component that runs and creates a lookup table that is used by the Resource Manager (RM) for establishing full duplex connections between resources.

With the addition of an IPT board into the system configuration, the lookup table is modified as follows:

Boards	Static Tx TDM Time Slots	Tx TDM Time Slots	Rx TDM Time Slots	TDM Time Slot Pool
DM/V2400A	1 to 240	**1 to 240**	**2001 to 2240**	---
IPT2400	241 to 480	241 to 480		
Third Party Boards	---	---	---	**2241 to 4096**

The time slot ranges in bold are reserved by the administration component of the application. If more IPT boards are included, additional transmit time slots are assigned to the IP Media resources. Note that for the IPT2400 board, the transmit (Tx) time slot range is 241 to 480 as statically assigned at board initialization, but these time slots can be changed at runtime using this feature.

Note: TDM Rx time slots for the Dialogic boards are always allocated dynamically by the application.

Use Case 1

This use case describes the validation of DTMF quality sent to and received from the 3G GSM Gateway in the figure above. The application allocates a voice resource (dxxB1C1) and IP Media resource (ipmB1C100) for establishing a media session between the 3G GSM Gateway and a test tool. Using this feature, the RM changes the transmit time slot of the ipmB1C100 device (which is statically allocated to time slot #341) to time slot #2001 to match the receive time slot of the voice resource (dxxB1C1), thereby establishing a full-duplex connection between the devices as shown in the following table
.

Device	Static Tx TDM Time Slot	Tx TDM Time Slot	Rx TDM Time Slot	Purpose
Voice device (dxxB1C1)	1	1	2001	Generate test DTMF Detect tones returned from SUT
IP Media device (ipmB1C100)	341	2001	1	Send/receive and encode/decode DTMF from AMR to PCM and PCM to AMR

Once the test is completed, the devices are closed and any transmit time slot allocations freed. Devices that are subsequently opened default to the static transmit time slot assignment, unless explicitly changed using the **ipm_SetParm( )** function.

Note: It is the responsibility of the application to keep track of the resource and time slot status.

This use case activities are as follows:

1. The user makes a request to set up a connection with a 3G GSM SUT to validate DTMF and audio quality.

2. The test application establishes the connection with the remote SUT using IPT (network) and IPM (media) resources.

3. The user selects the DTMF tones to send and the test application generates and sends DTMF tones to the SUT. The tones are generated using a DM/V2400A voice resource and G.711 encoding.

4. The DTMF is received by the IPM resource listening to transmit time slot #1 of the DM/V2400A board voice resource.

5. The IPM resource encodes the PCM data stream to AMR and sends the RTP packet to the SUT.

6. The SUT receives the RTP stream and interprets the payload as a DTMF tone.

7. The SUT transmits the received DTMF back to the IPM resource in AMR format.

8. The IPM resource transcodes the AMR RTP stream to PCM which is then transmitted on time slot #2001 to the DM/V2400A board voice resource for tone detection.

9. The DM/V2400A board voice resource signals the application that a tone has been received.

Using this scenario, the user can verify if the 3G GSM Gateway is receiving, detecting, and generating DTMF correctly.

Use Case 2

The same scenario as use case 1 above can be performed between the 3G GSM Gateway IPM and voice resources to verify voice quality. The customer can use a recorded voice file, which is sent to the SUT. Then, some form of perceptual measurement (PESQ or PSQM) can be used to evaluate the voice quality of the recorded voice received versus that which was sent.

Use Case 3

Another use case for this feature is the verification of the behavior of a specialized switch when receiving specific tones. These tests are typically run remotely to ensure the switch is recognizing the tones and responding appropriately. The tones typically tested are call progress, SIT, and fax tones.

1.3.4 Using This Feature

The following is the sequence of function calls involved when using this feature:

1. **ipm_Open( )** - Opens an IP Media device and returns a valid handle. The transmit time slot is allocated to the static time slot number.

2. **ipm_SetParm( )** - Sets the transmit time slot to a specific number. This is achieved by setting up an IPM_PARM_INFO structure where the eParm field is PARMCH_TX_TIMESLOT and the *pvParmValue field is a pointer to the parameter value, that is, the time slot number to be allocated. See the **ipm_SetParm( )** function reference page in the *IP Media Library API Library Reference* for more detail.

3. **ipm_GetParm( )** (optional) - Validates that the time slot has been changed to the desired time slot. This can be used at any time to get the active transmit time slot of an IP Media device. This functionality is equivalent to that provided by the **ipm_GetXmitSlot( )** function.

4. **ipm_Close( )** - Closes an IPM device deallocating all time slots. This should be the final step before the resource is returned to the free resource pool.

***Caution**: Assigning transmit time slots incorrectly can cause physical damage on the IPT or any other H.110 board when two or more devices are transmitting on the same TDM time slot. It is the responsibility of the application to ensure that the transmit time slot of each IP Media device is allocated to a time slot that has no other device transmit time slots allocated to it.*

Notes:1. This feature is applicable to IPT boards only.

2. The default static time slot assignment applies if not explicitly changed using this feature.

3. It is assumed that the system is operating in Passive mode, where the customer application is responsible for the establishment and management of H.110 clocking (primary clock, signal reference, and any associated fallback strategies, respectively).

4. When a single board stop operation is performed (that is, an IPT board is stopped without stopping the system), all dynamically assigned transmit time slots associated with the IP Media devices on the IPT board are deallocated.

- If the IPT board is restarted without any other boards being stopped prior to the restart, the same set of static time slots are allocated to the IPT board.

- If multiple boards are stopped, then the static time slot allocations may change depending on the order in which the boards are restarted.

5. It is the responsibility of the application to ensure that a time slot value specified is in the range 0 to 4095. The **ipm_SetParm( )** and **ipm_GetParm( )** functions do not perform any validation on the value specified.

6. The **ipm_GetXmitSlot( )** function, which returns the currently allocated transmit time slot for an IP Media device, has not changed as a result of this feature.

7. Closing an IP Media device, then reopening it, resets the transmit time slot back to the static time slot allocated during board initialization (firmware download).

1.3.5 Documentation

The online bookshelf provided with System Release 6.1 for Linux contains information about all system release features including features for application development, configuration, administration, and diagnostics.

For more information about the IP Media Library API, see the following documents:

- *IP Media Library (IPML) API Programming Guide*
- I*P Media Library API for Linux and Windows Library Reference*

Note: The online bookshelf has not been updated for this feature, so the manuals above do not contain information relating to this feature.

1.4 Runtime Control of Double Answer for R2MF

With this Service Update, a connection method called double answer is now supported for rejecting collect calls on a call-by-call basis.

1.4.1 Feature Description

Currently, double answer signaling is statically enabled or disabled by setting the **CDP_DOUBLE_ANSWER_FLAG** parameter in the CDP file. However, this setting applies to all the calls on the channels and cannot be controlled on a call-by-call basis.

With this new feature, double answer can be triggered on a call-by-call basis by issuing **gc_AnswerCall( )** with the number of rings ORed with a new define, **GC_DBL_ANSWER** (0x100).

Note: The double answer feature must be disabled (disabled by default) in the CDP file. If the double answer feature is enabled by setting the CDP_DOUBLE_ANSWER_FLAG parameter in the CDP file, then there will be no application control of this feature on a call-by-call basis (this feature will always be triggered).

Note: If **gc_AnswerCall( )** is issued with the number of rings ORed with **GC_DBL_ANSWER** on a protocol that does not support double answer functionality, there will be no error reported as there is no range checking being done in the PDK protocols for the number of rings. The expected behavior is that while the inbound side is busy generating the ring back tone (>= 256 rings), the remote side will time out and the call will eventually get dropped.

1.4.2 Supported Boards

DM3

- Intel NetStructure® DM/V-A Combined Media Boards
- Intel NetStructure® DM/V-B Combined Media Boards

Springware

- Intel® Dialogic® D/300JCT-E1 Combined Media Boards
- Intel® Dialogic® D/600JCT-1E1 Combined Media Boards
- Intel® Dialogic® D/600JCT-2E1 Combined Media Boards

1.4.3 Example Code

```
#include <stdio.h>
#include <srllib.h>
#include <gclib.h>
#include <gcerr.h>
/*
* Assume the following has been done:
* 1. Opened line devices for each time slot on DTIB1.
* 2. Wait for a call using gc_WaitCall()
* 3. An event has arrived and has been converted to a metaevent
* using gc_GetMetaEvent() or gc_GetMetaEventEx() (Windows)
* 4. The event is determined to be a GCEV_OFFERED event
*/
```

```
int answer_call(int num_rings, int dbl_answ_flag)
{
       CRN crn; /* call reference number */
       GC_INFO gc_error_info; /* GlobalCall error information data */
       int rings = 0;

       /*
       * Do the following:
       * 1. Get the CRN from the metaevent
       * 2. Proceed to answer the call as shown below
       */
       crn = metaevent.crn;

       /*
       * Answer the incoming call. Check the dbl_answ_flag to determine
       * if double answer should be triggered or not
       */
       if (dbl_answ_flag)
           rings = num_rings | GC_DBL_ANSWER;
       else
           rings = num_rings;

       if (gc_AnswerCall(crn, rings, EV_ASYNC) != GC_SUCCESS) {
           /* process error return as shown */
           gc_ErrorInfo( &gc_error_info );
           printf ("Error: gc_AnswerCall() on device handle: 0x%lx, GC ErrorValue: 0x%hx - %s,
               CCLibID: %i - %s, CC ErrorValue: 0x%lx - %s\n",
               metaevent.evtdev, gc_error_info.gcValue, gc_error_info.gcMsg,
               gc_error_info.ccLibId, gc_error_info.ccLibName,
               gc_error_info.ccValue, gc_error_info.ccMsg);
           return (gc_error_info.gcValue);
       }

       /*
       * gc_AnswerCall() terminates with GCEV_ANSWERED event
       */
       return (0);
}
```

1.4.4 Documentation

The online bookshelf provided with System Release 6.1 for Linux contains information about all system release features including features for application development, configuration, administration, and diagnostics.

For more information about the Global Call API, see the following documents:

- *Global Call API for Linux and Windows Operating Systems Library Reference*
- *Global Call API for Linux Operating Systems Programming Guide*

For features specific to E1 (R2) technology, see the following documents:

- *Global Call E1/T1 CAS/R2 Technology User's Guide for Linux and Windows*
- *Global Call Country Dependent Parameters (CDP) for PDK Protocols Configuration Guide*

1.5 Support for D/4PCI Board

With this Service Update, the D/4PCI Voice Board that was supported in older system releases is now supported in System Release 6.1 for Linux. In support of this board, you may now select D/4PCI from the Model Names list when using the configuration manager for Linux.

When configuring Springware boards, boards that share the same Device ID with other boards are displayed in the Board Summary screen as “Name Unresolved.” When you select the Thumb Wheel number of the board to configure it, a menu for that board is displayed. When you choose the Model Name option, a list of possible model names for the board are displayed. D/4PCI now appears in the list with these other model names:

D/41JCT
VFX/41JCT
D/4PCI-U
D/4PCI

The detailed procedures for using the configuration manager for Linux to configure boards are described *Intel® Springware Architecture Products on Linux Configuration Guide.*

Documentation

The online bookshelf provided with System Release 6.1 for Linux contains information about all system release features including features for application development, configuration, administration, and diagnostics.

For more information about configuring boards, see the following document:

- *Intel® Springware Architecture Products on Linux Configuration Guide*

1.6 Support for PCI Express Boards

The new Intel® Dialogic® D/120JCT-LS-EW combined media board is now supported in System Release 6.1 for Linux. The D/120JCT-LS-EW is a 12-port analog telecom board in a PCI Express form factor. The D/120JCT-LS-EW supports voice, fax, and software-based speech recognition processing in a single PCI Express slot, and provides 12 analog telephone interface circuits for direct connection to analog loop start lines.

Note: The D/120JCT-LS-EW board implements a PCI Express x1 lane configuration, allowing it to be used in any full-length PCI Express slot that fully meets PCI Express Card Electromechanical Specification Revision 1.0a or higher. Up-plugging the board into a slot with a x4 or larger link connector will help ensure adequate power budgeting for boards with power requirements of 10W or more, and is therefore recommended for such boards. For further information, see the Installation Guide (Quick Install Card) that is provided with the board.

1.7 Support for 12 GB RAM

With this Service Update, System Release 6.1 for Linux now supports 12 GB RAM.

1.8 Support for Reporting Billing Type

With this Service Update, for DM3 boards, there is now a way for the application to know which billing type (for a call on PDK R2 protocol) was received when the lines are available for call establishment. B tones are sent to indicate whether the line is available or not, and also to indicate the type of billing for the call (for example, CHARGE, NO CHARGE, or CHARGE WITH CLEARING FROM INBOUND).

This feature is already supported on Springware boards; however, CHARGE WITH CLEARING FROM INBOUND is a new billing type that is also supported on Springware now.

1.8.1 Feature Description

The user is notified of the billing type for a successful call establishment. The **gc_GetCallInfo( )** function with info_id equal to CALLINFOTYPE is used to retrieve the billing type. The following mappings are implemented:

Group B Tone	Billing Type String Returned
GrpB - line free, charge	“CHARGE”
GrpB - line free, no charge	“NO CHARGE”
GrpB - line free, charge with clearing from inbound only	“CHARGE WITH CLEARING FROM INBOUND”

For B tones indicating unavailability of the line (call establishment failure), the following mappings are used for assigning cause values to the GCEV_DISCONNECT event:

Group B Tone	GC Cause Value	Description
GrpB - User Busy	GCRV_BUSY	“Line is busy”
GrpB - Network Congestion	GCRV_CONGESTION	“Congestion”
GrpB - Normal Clearing	GCRV_NORMAL	“Normal Clearing”
GrpB - UnAssigned Number	For DM3: GCRV_UNALLOCATED For Springware: GCRV_NOT_INSERVICE	For DM3: “Number not allocated” For Springware: “Number not in service”
GrpB - SIT	For DM3: GCRV_SIT_UNKNOWN For Springware: GCRV_CEPT	For DM3: “Unknown SIT detected” For Springware: “Operator intercept”
GrpB - Rejected	GCRV_REJECT	“Call Rejected”

Note: If the billing type is not supported on a protocol, then **gc_GetCallInfo(CALLINFOTYPE)** returns “UNKNOWN BILLING”.

1.8.2 Supported Boards

DM3

- Intel NetStructure® DM/V-A Combined Media Boards
- Intel NetStructure® DM/V-B Combined Media Boards

Springware

- Intel® Dialogic® D/300JCT-E1 Combined Media Boards
- Intel® Dialogic® D/600JCT-1E1 Combined Media Boards
- Intel® Dialogic® D/600JCT-2E1 Combined Media Boards

1.8.3 Documentation

The online bookshelf provided with System Release 6.1 for Linux contains information about all system release features including features for application development, configuration, administration, and diagnostics.

For more information about the Global Call API, see the following documents:

- *Global Call API Library Reference*
- *Global Call API Programming Guide*

For features specific to E1 (R2) technology, see the following documents:

- *Global Call E1/T1 CAS/R2 Technology Guide*
- *Global Call Country Dependent Parameters (CDP) for PDK Protocols Configuration Guide*

1.9 Dynamically Adding and Deleting SS7 Circuit Groups

With the Service Update, the user can program a Global Call SS7 application to dynamically add new and delete existing circuit groups at runtime. Previously, the Global Call SS7 library only supported static configuration which meant that all parameters set in the configuration files (*config.txt* and *gcss7.cfg*) were not allowed to be modified using Global Call APIs once the system started. By being able to dynamically add and delete circuit groups, the user can start with an initial number of trunks in the system, and then enable more trunks gradually without restarting the system and application. The user also has the option to delete circuit groups.

1.9.1 Feature Description

The ability to dynamically add and delete SS7 circuit groups at runtime is provided by the **gc_SetConfigData( )** function. To use **gc_SetConfigData( )** to add or delete a circuit group, the following have been developed and added to the *Libgcs7.h* file:

- For adding circuit groups, a new set ID: SS7SET_ADD_CCTGRP with the following Parm IDs:
  - SS7PARM_GRP_ID
  - SS7PARM_ISUP_CFG_GRP
  - SS7PARM_TRUNK_CFG

  Note: For the addition of a new circuit group, the interface includes the same information as required by the ISUP_CFG_CCTGRP configuration command in the *config.txt* file; that is, group name, trunk name, and optional data.
- For deleting circuit groups, a new set ID: SS7SET_END_CCTGRP with the following Parm ID:
  - SS7PARM_GRP_ID.
- Three new data structures.

1.9.2 Supported Boards

- Intel NetStructure® SS7HDPD4TE
- Intel NetStructure® SS7HDCN16
- Intel NetStructure® SS7HDCS8
- Intel NetStructure® SS7HDCD16
- Intel NetStructure® SS7HDCQ16
- Intel NetStructure® DMT160
- Intel NetStructure® DMVB
- Intel NetStructure® DMVA

1.9.3 New Set IDs and Parm IDs

The following new defines have been added in *Libgcs7.h* .

Adding A Circuit Group

Set ID:

#define **SS7SET_ADD_CCTGRP** (0x4501 or 17665)

Parm IDs:

- #define **SS7PARM_GRP_ID** (0x1) - Must be group ID (integer value)
- #define **SS7PARM_ISUP_CFG_GRP** (0x2) - Mandatory for adding or ending a circuit group, which provides the basic data required by ISP_MSG_CNF_GRP message. Its value type is GCSS7_ISUP_CFG_CCTGRP data structure.

- #define **SS7PARM_TRUNK_CFG** (0x3) - Mandatory for adding a circuit group, which provides the data required by Global Call SS7 to configure a trunk device. Its value type is GCSS7_TRUNK_CFG data structure.

Deleting A Circuit Group

Set ID:

#define **SS7SET_DEL_CCTGRP** (0x4502 or 17666)

Parm IDs:

- #define **SS7PARM_GRP_ID** (0x1) - Must be group ID (integer value). There is no other data required by ISP_MSG_END_GRP message.

1.9.4 New Data Structures

The following new structure definitions have been added in the *Libgcs7.h* file.

- GCSS7_ISUP_CFG_CCTGRP
- GCSS7_TRUNK_CFG
- GCSS7_ISUP_CCTGRP_EXTRA

The new data structures are described below.

GCSS7_ISUP_CFG_CCTGRP

```
typedef struct {
    unsigned long   version;
    unsigned long   dpc;
    unsigned short  basic_cic;
    unsigned short  basic_cid;
    unsigned long   cic_mask;
    unsigned long   options;
    unsigned char   user_inst;
    unsigned char   user_id;
    unsigned long   opc;
    unsigned long   ssf;
    unsigned char   variant;
    unsigned long   option2;
} GCSS7_ISUP_CFG_CCTGRP, *GCSS7_ISUP_CFG_CCTGRPP;
```

Description

The following GCSS7_ISUP_CFG_CCTGRP data structure is designed for configuration of an ISUP circuit group, which matches the ISUP_CFG_CCTGRP configuration command in *config.txt* file.

Field Descriptions

The fields of the GCSS7_ISUP_CFG_CCTGRP data structure are described as follows:

version
: The version of the data structure. The initial version is 0x1000.

dpc
: The Destination Point Code (DPC) for all circuits in the circuit group.

basic_cic
: The first Circuit Identification Code (CIC) in the circuit group.

basic_cid
: The Logical Circuit Identifier (CID) corresponding to the first CIC.

cic_mask
: A 32-bit mask with bits set to indicate which circuits are to be allocated.

options
: A 32-bit value containing run-time options for the ISUP circuit group.

user_inst
: The instance number of the user application.

user_id
: The user application module ID.

opc
: The Originating Point Code (OPC) for all circuits in the circuit group.

ssf
: The value to be used in the Sub_Service Field (SSF) of all ISUP messages.

variant
The protocol variant for this circuit group.

option2
A 32-bit value containing additional run-time options for the ISUP circuit group.

GCSS7_TRUNK_CFG

```
typedef struct {
    unsigned long   version;
    char            trunk_name[20];
    unsigned char   base_ts;
    unsigned char   pref_siu;
} GCSS7_TRUNK_CFG, *GCSS7_TRUNK_CFGP;
```

Description

The following GCSS7_TRUNK_CFG data structure is designed for configuration of a Global Call trunk device on the circuit group, which matches the CGrp configuration command in *gcss7.cfg* file.

Field Descriptions

The fields of the gCSS7_TRUNK_CFG data structure are described as follows:

version
: The version of the data structure. The initial version is 0x1000.

trunk_name[20]
: The physical device name where the circuits in the group are terminated (e.g., dtiB1 or dkB1).

base_ts
: The first time slot of the trunk that corresponds the first circuit of the group.

pref_siu
: The default SIU for the group.

Add a Circuit Group

In **gc_SetConfigData( )** function, set all arguments as follows:

- target_type = GCTGT_CCLIB_SYSTEM
- target_id = GC_SS7_LIB or 5
- target_datap = GC_PARM_BLKP parameter pointer, as constructed by the utility function **gc_util_insert_parm_ref( )** or **gc_util_insert_parm_val( )** for configuration of circuit groups.

To add a circuit group, do the following:

1. Call **gc_util_insert_parm_val( )** to insert {SS7SET_ADD_CCTGRP, SS7PARM_GRP_ID} with integer value: GroupID

2. Call **gc_util_insert_parm_ref( )** to insert {SS7SET_ADD_CCTGRP, SS7PARM_ISUP_CFG_GRP} with data structure: GCSS7_ISUP_CFG_CCTGRP

3. Call **gc_util_insert_parm_ref( )** to insert {SS7SET_ADD_CCTGRP, SS7PARM_TRUNK_CFG} with data structure: GCSS7_TRUNK_CFG

- time_out = time interval (in seconds) during which the parameter value must be updated. If the interval is exceeded, the update request is ignored. This parameter is supported in synchronous mode only, and it is ignored when set to 0.
- update_cond = ignored in Global Call SS7.
- request_idp = pointer to the location for storing the request ID, output from Global Call.
- mode = EV_SYNC for synchronous execution.

Sample Code

```
#include <stdio.h>
#include <srllib.h>
#include <gclib.h>
#include <gcerr.h>
#include <Libgcs7.h>

int AddCircuitGrp(int a_GroupID, unsigned long a_DPC, unsigned long a_OPC,
                      unsigned short a_FirstCIC, unsigned short a_FirstCID,
                      char * a_TrunkName, long * a_pRequestID)
{
    GC_PARM_BLK * t_pParmBlk = NULL;
    GCSS7_ISUP_CFG_CCTGRP t_GrpCfg;
    GCSS7_TRUNK_CFG t_TrunkCfg;
    int t_result = 0;

    *a_pRequestID = 0;
    /* Initialize the GCSS7_ISUP_CFG_CCTGRP data structure */
    memset(&t_GrpCfg, 0, sizeof(GCSS7_ISUP_CFG_CCTGRP));
    t_GrpCfg.dpc = a_DPC;
    t_GrpCfg.basic_cic = a_FirstCIC;
    t_GrpCfg.basic_cid = a_FirstCID;
    t_GrpCfg.cic_mask = 0x7fff7fff;
    t_GrpCfg.options = 0x071e;
    t_GrpCfg.user_inst = 0;
```

```
    t_GrpCfg.user_id = 0x4d;
    t_GrpCfg.opc = a_OPC;
    t_GrpCfg.ssf = 0x8;
    t_GrpCfg.variant = 0;
    t_GrpCfg.options2 = 0;

    /* Initialize the GCSS7_TRUNK_CFG data structure */
    memset(&t_TrunkCfg, 0, sizeof(GCSS7_TRUNK_CFG));
    strcpy( t_TrunkCfg.trunk_name, a_TrunkName );
    t_TrunkCfg.base_ts = 1;  /* Started from the first timeslot */

    /* Insert the Group ID */
    t_result = gc_util_insert_parm_val(&t_pParmBlk, SS7SET_ADD_CCTGRP, SS7PARM_GRP_ID,
sizeof(int), a_GroupID);
    if (t_result)
        {
            /* Process error */
        return t_result;
    }
    /* Insert the parameter SS7PARM_ISUP_CFG_GRP */
    t_result = gc_util_insert_parm_ref(&t_pParmBlk, SS7SET_ADD_CCTGRP, SS7PARM_ISUP_CFG_GRP,
sizeof(GCSS7_ISUP_CFG_CCTGRP), &t_GrpCfg);
    if (t_result)
    {
        /* Process error */
        return t_result;
    }
    /* Insert the parameter SS7PARM_TRUNK_CFG */
    t_result = gc_util_insert_parm_ref(&t_pParmBlk, SS7SET_ADD_CCTGRP, SS7PARM_TRUNK_CFG,
sizeof(GCSS7_TRUNK_CFG), &t_TrunkCfg);
    if (t_result)
    {
        /* Process error */
        return t_result;
    }
    /* Add a new circuit group with a_GroupID */
    t_result  = gc_SetConfigData(GCTGT_CCLIB_SYSTEM, GC_SS7_LIB, t_pParmBlk, 0,
GCUPDATE_IMMEDIATE, a_pRequestID, EV_SYNC);
    if (t_result)
    {
        /* Process the error */
        gc_util_delete_parm_blk(t_pParmBlk);
        return t_result;
    }
    gc_util_delete_parm_blk(t_pParmBlk);
    return t_result;
}
```

1.9.5 Delete a Circuit Group

In **gc_SetConfigData( )** function, set all arguments as follows:

- target_type = GCTGT_CCLIB_SYSTEM
- target_id = GC_SS7_LIB or 5
- target_datap = GC_PARM_BLKP parameter pointer, as constructed by the utility function **gc_util_insert_parm_ref( )** or **gc_util_insert_parm_val( )** for configuration of circuit groups.

To delete a circuit group, call **gc_util_insert_parm_val( )** to insert {SS7SET_DEL_CCTGRP, SS7PARM_GRP_ID} with integer value: GroupID

- time_out = time interval (in seconds) during which the parameter value must be updated. If the interval is exceeded, the update request is ignored. This parameter is supported in synchronous mode only, and it is ignored when set to 0.
- update_cond = ignored in Global Call SS7
- request_idp = pointer to the location for storing the request ID, output from Global Call
- mode = EV_ASYNC for asynchronous execution

Sample Code

```
#include <stdio.h>
#include <srllib.h>
#include <gclib.h>
#include <gcerr.h>
#include <Libgcs7.h>

int DeleteCircuitGrp(int a_GroupID, long * a_pRequestID)
{
    GC_PARM_BLK * t_pParmBlk = NULL;
    int t_result = 0;

    *a_pRequestID = 0;
    /* Insert the Group ID */
    t_result = gc_util_insert_parm_val(&t_pParmBlk, SS7SET_DEL_CCTGRP, SS7PARM_GRP_ID,
sizeof(int), a_GroupID);
    if (t_result)
    {
        /* Process error */
        return t_result;
    }
    /* Delete a circuit group with a_GroupID */
    t_result = gc_SetConfigData(GCTGT_CCLIB_SYSTEM, GC_SS7_LIB, t_pParmBlk, 0,
GCUPDATE_IMMEDIATE, a_pRequestID, EV_ASYNC);
    if (t_result)
    {
        /* Process the error */
        gc_util_delete_parm_blk(t_pParmBlk);
        return t_result;
    }
    gc_util_delete_parm_blk(t_pParmBlk);
    return t_result;
}

int main()
{
int t_RequestID = 0;
AddCircuitGrp(1, 555, 777, 1, 1,"dkB1", &t_RequestID);

DeleteCircuitGrp(1, &t_RequestID);
}
```

1.9.6 Application Development Guidelines

In order to dynamically add or delete existing circuit groups at runtime, consider the following guidelines:

- The boards (SS7 or DM3) representing new circuit groups should be downloaded successfully.

- The total number of circuit groups or the total number of circuits in the SS7 board system (including static and dynamically allocated) can not exceed <num_grps> or <num_ccts> of **ISUP_CONFIG** defined in *config.txt*.
- All T1/E1 LIUs of SS7 boards in the system to be used must be fully configured in *config.txt* (using the LIU_CONFIG command).
- The circuit group to be deleted has to exist in the system.
- All line devices (including trunk device) on that circuit group must be closed by using using **gc_Close( )** before the circuit group can be deleted.
- All MPT links of SS7 board to be used have to be configured in *config.txt* (using MTP_LINKSET and MTP_LINK commands).
- Modifying the existing circuit groups is not supported, which includes adding (or disabling) individual circuits.
- Dynamic configuration of Global Call SS7 circuit groups is only supported in the SS7 board system, not the SIU system.
- Dynamic configuration of circuit groups does not include updating the configuration files (*config.txt* and *gcss7.cfg*).
- Each **gc_SetConfigData( )** function call only allows the user application to add or delete one circuit group at a time.
- Dynamic configuration of LIUs and SCbus route is not supported.
- The maximum number of circuits in a circuit group should not exceed 31 for E1 trunk or 24 for T1 trunk.
- The maximum number of circuit groups (including both original and dynamically added) is limited to 64 E1/T1 trunks.
- The configuration remains persistent as long as the SS7 service/daemon is running. If only the Global Call SS7 application is stopped, the newly added or deleted circuit group still remains configured.

1.9.7 Documentation

The online bookshelf provided with System Release 6.1 for Linux contains information about all system release features including features for application development, configuration, administration, and diagnostics.

For more information about the Global Call API in general, see the following documents:

- *Global Call API Programming Guide*
- *Global Call API Library Reference*

For features specific to SS7 technology, see:

- *Global Call SS7 Technology Guide*

1.10 Global Call Support for Time Slots on SS7 Boards Running in DTI Mode

With the Service Update, Global Call works with SS7 boards that include trunks not configured for SS7 signalling (DTI mode); i.e., all the time slots on these trunks operate in clear channel mode.

1.10.1 Feature Description

The SS7 boards are supported in a non SS7 signaling environment. This allows the application to use an SS7 board (for example, SS7HDCN16) in a clear channel mode to terminate E1/T1 trunks and switch them over the CT bus.

Supported Boards

- Intel NetStructure® SS7HDCN16
- Intel NetStructure® SS7HDCS8
- Intel NetStructure® SS7HDCD16
- Intel NetStructure® SS7HDCQ16

Configuration

You modify three configuration files to use this feature:

- *gcss7.cfg*
- *config.txt*
- *system.txt*

Global Call SS7 Software Configuration (gcss7.cfg)

The Global Call software configuration file (*gcss7.cfg*) is modified to specify a new **ClearGrp** parameter that includes the following fields:

<trunk_name>

The virtual device name of the trunk (for example, dkB1)

Note: This specifies the physical device where the circuits in the group are terminated. The <trunk_name> refers to one of the trunks on an Intel NetStructure SS7 board, where dkB1 is the name of the first trunk (first LIU defined in the *config.txt* file), dkB2 is the name of the second trunk (second LIU defined in the *config.txt* file), and so on. The same name is used as a basis by the application for the network device name when it opens a Global Call SS7 device.

<ts_mask>

Specifies the time slots that are to be used as clear channels. Each bit in this mask corresponds to a physical time slot on the trunk where a “1” indicates that the time slot

is to be used in clear channel mode. For example, 0x7fffffff for E1 or for T1 for 0xffffff indicates that all time slots in the trunk are to be used in clear channel mode.

For example:

```
# Clear Channel Group configuration.”
# ClearGrp <"trunk_name"> <ts_mask>
ClearGrp dkB1 0x7fffffff
ClearGrp dkB2 0x7fffffff
ClearGrp dkB3 0x7fffffff
ClearGrp dkB4 0x7fffffff
```

config.txt

In clear channel mode, the minimum configuration required is the board and LIU configuration in the *config.txt* file, as shown in the example below for an SS7HDC board:

```
* For SS7HD cP boards:
* SS7_BOARD <board_id> <board_type> <flags> <code_file> <run_mode>
SS7_BOARD  0 SS7HDC 0x00C2  ss7.dc4  dti
*
* LIU_CONFIG <board_id> <liu_id> <liu_type> <line_code> <frame_format> <crc_mode>
LIU_CONFIG 0 0 5 1 1 1
LIU_CONFIG 0 1 5 1 1 1
LIU_CONFIG 0 2 5 1 1 1
LIU_CONFIG 0 3 5 1 1 1
```

system.txt

The following is an example of the minimum configuration required in the *system.txt* file for clear channel operation (i.e, when there is no SS7 signaling supported):

```
*
* Essential modules running on host:
*
LOCAL           0x20            * ssd/ssds/ssdh - Board interface task
LOCAL           0x00            * tim_nt - Timer task
*
* Optional modules running on the host:
*
LOCAL           0xcf            * s7_mgt - Management/config task
LOCAL           0xef            * s7_log - Display and logging utility
LOCAL           0x4d            * GCSS7
*
*
* Essential modules running on the board (all redirected via ssd):
*
REDIRECT        0x10    0x20    * CT bus/Clocking control module
REDIRECT        0x8e    0x20    * On-board management module
*
*

* Redirection of status indications:
*
*REDIRECT       0xdf    0xef    * LIU/MTP2 status messages -> s7_log
*REDIRECT        0xdf    0x4d    * LIU/MTP2 status messages -> GCSS7
*REDIRECT       0xef    0x4d    * trace messages -> GCSS7
*
* Now start-up all local tasks:
*   (For PCCS6 start-up use ssd.exe and ssd_poll.exe,
*   for SPCI4/SPCI2S/CPM8 start-up use ssds.exe and
```

```
*   for SS7HD boards use ssdh.exe)
*
* FORK_PROCESS  ssd.exe
* FORK_PROCESS  ssd_poll.exe
* FORK_PROCESS  ssdh.exe
FORK_PROCESS    ./ssds -d
FORK_PROCESS    ./tim_lnx
FORK_PROCESS    ./tick_lnx
FORK_PROCESS    ./s7_mgt -d -i0x4d
FORK_PROCESS    ./s7_log
* FORK_PROCESS  upe.exe
```

Additional Configurations

In addition to the configuration for clear channel (DTI mode), there can be additional configurations for SS7 signaling on other SS7 boards in the system. The following are examples of a mixed configuration (SS7 signaling on a SS7HDCS8 board and clear channels on SS7HDCN16 board):

```
*********************************************************************
* Example Protocol Configuration File (config.txt) for use with
* Intel(R) NetStructure(TM) SS7 Boards.
*
* Boards supported are PCCS6, SPCI4, SPC2S, CPM8 and the SS7HD range.
* (note, not all boards are supported on all operating systems).
*
* This file needs to be modified to suit individual circumstances.
* Refer to the relevant Programmer's Manuals for further details.
*
*****************************************************************************
* For SS7HD cP boards:
* SS7_BOARD <board_id> <board_type> <flags> <code_file> <run_mode>
SS7_BOARD  0 SS7HDC 0x0042  ss7.dc4  ISUP * Master -- HDC
SS7_BOARD  1 SS7HDC 0x00C2  ss7.dc4  dti

* Configure individual E1/T1 interfaces:
* LIU_CONFIG <board_id> <liu_id> <liu_type> <line_code> <frame_format>
*            <crc_mode>
LIU_CONFIG  0  0  4  4  7  4
LIU_CONFIG  0  1  4  4  7  4
LIU_CONFIG  0  2  4  4  7  4
LIU_CONFIG  0  3  4  4  7  4
LIU_CONFIG  1  0  4  4  7  4
LIU_CONFIG  1  1  4  4  7  4
LIU_CONFIG  1  2  4  4  7  4
LIU_CONFIG  1  3  4  4  7  4
LIU_CONFIG  1  4  4  4  7  4
LIU_CONFIG  1  5  4  4  7  4
LIU_CONFIG  1  6  4  4  7  4
LIU_CONFIG  1  7  4  4  7  4
LIU_CONFIG  1  8  4  4  7  4
LIU_CONFIG  1  9  4  4  7  4
LIU_CONFIG  1  10  4  4  7  4
LIU_CONFIG  1  11  4  4  7  4
LIU_CONFIG  1  12  4  4  7  4
LIU_CONFIG  1  13  4  4  7  4
LIU_CONFIG  1  14  4  4  7  4
LIU_CONFIG  1  15  4  4  7  4

* MTP_CONFIG <reserved> <reserved> <options>
MTP_CONFIG  0  0  0x00040f00
```

```
* MTP_LINKSET <linkset_id> <adjacent_spc> <num_links> <flags> <local_spc> <ssf>
MTP_LINKSET  0  777  1  0x0000  555  0x03 * Loopback - HDC 1
MTP_LINKSET  1  555  1  0x0000  777  0x03 * Loopback - HDC 2

* MTP_LINK <link_id> <linkset_id> <link_ref> <slc> <board_id> <blink>
*          <stream> <timeslot> <flags>
* Note 1: For PCCS6 boards the first LIU port is stream=16 whilst for other
* boards the first LIU port is stream=0.
* Note 2: The SS7HD board requires a compound parameter for blink of the form
* sp_id-sp_channel.
*
* For SS7HD boards:
MTP_LINK  0  0  0  0  0  0-0  0   24  0x0006 * LIU 0, signalling on TS 24, for loopback
MTP_LINK  1  1  0  0  0  0-1  1   24  0x0006 * LIU 1, signalling on TS 24, for loopback
*
* Define a route for each remote signaling point:
* MTP_ROUTE <dpc> <linkset_id> <user_part_mask>
MTP_ROUTE  777  0  0x0020  0x0000  0
MTP_ROUTE  555  1  0x0020  0x0000  0
*

* ISUP parameters:
*
* Configure ISUP module:
* ISUP_CONFIG <reserved> <reserved> <user_id> <options> <num_grps> <num_ccts>
ISUP_CONFIG  0  0  0x4d  0x0774  2  50

*
* Configure ISUP circuit groups:
* ISUP_CFG_CCTGRP <gid> <dpc> <base_cic> <base_cid> <cic_mask> <options>
*                       <user_inst> <user_id> <opc> <ssf> <variant> <options2>
ISUP_CFG_CCTGRP  0  777  0x01  0x00  0x7fffff 0x0000401c  0  0x4d  555  0x3  2  0x00
ISUP_CFG_CCTGRP  1  555  0x01  0x19  0x7fffff 0x0000401c  0  0x4d  777  0x3  2  0x00



**********************************************************************
*
* Example System Configuration File (system.txt) for use with the
* Windows Development Package for Intel(R) NetStructure(TM) SS7 Boards
*
* Edit this file to reflect your configuration.
*
**********************************************************************
*
* Essential modules running on host:
*
LOCAL           0x20            * ssd/ssds/ssdh - Board interface task
LOCAL0x00* tim_nt - Timer task
*
* Optional modules running on the host:
*
LOCAL0xcf* s7_mgt - Management/config task
LOCAL           0xef            * s7_log - Display and logging utility

*LOCAL0x2d* upe - Example user part task
LOCAL0x4d* GC SS7 Service
*
* Modules that optionally run on the host
*
*LOCAL 0x23* ISUP module
*LOCAL 0x4a* TUP module
*LOCAL 0x33* SCCP module
```

```
*LOCAL 0x14* TCAP module
*LOCAL0x22* MTP3 module
*
* Essential modules running on the board (all redirected via ssd):
*

REDIRECT0x710x20* MTP2 module (except SS7HD boards)
REDIRECT0x81    0x20    * MTP2 module_id for SP 0 (SS7HD boards only)
REDIRECT0x91    0x20    * MTP2 module_id for SP 1 (SS7HD boards only)
REDIRECT0xe1    0x20    * MTP2 module_id for SP 2 (SS7HD boards only)
REDIRECT0xf1    0x20    * MTP2 module_id for SP 3 (SS7HD boards only)
REDIRECT        0x10    0x20    * CT bus/Clocking control module
REDIRECT0x8e0x20* On-board management module
*
* Modules that optionally run on the board (all redirected via ssd):
*

REDIRECT 0x230x20* ISUP module
* REDIRECT 0x4a0x20* TUP module
* REDIRECT 0x330x20* SCCP module
* REDIRECT 0x140x20* TCAP module
REDIRECT0x220x20* MTP3 module
*
* Redirection of status indications:
*
REDIRECT0xdf0x4d* LIU/MTP2 status messages -> GCSS7
*REDIRECT0xef0x4d* other/trace -> GCSS7
REDIRECT0xdf0xef* LIU/MTP2 status messages -> s7_log
*
* Now start-up all local tasks:
*   (For PCCS6 start-up use ssd.exe and ssd_poll.exe,
*   for SPCI4/SPCI2S/CPM8 start-up use ssds.exe and
*   for SS7HD boards use ssdh.exe)
*
* FORK_PROCESSssd.exe
* FORK_PROCESSssd_poll.exe
* FORK_PROCESSssds.exe
FORK_PROCESSssdh.exe -d
FORK_PROCESSStim_nt.exe
FORK_PROCESSStick_nt.exe
FORK_PROCESSs7_mgt.exe -d -i0x4d
FORK_PROCESSs7_log.exe
* FORK_PROCESSupe.exe
*
*
*****************************************************************************
```

1.10.2 Feature Limitations, Caveats and Restrictions

- Opening of trunk devices, for example dkB1, is not supported.
- The following APIs are supported in clear channel mode:
  - **gc_Attach( )**
  - **gc_AttachResource( )**
  - **gc_Close( )**
  - **gc_Detach( )**
  - **gc_GetNetworkH( )**
  - **gc_GetResourceH( )**
  - **gc_GetXmitSlot( )**
  - **gc_Listen( )**

- **gc_OpenEx( )**
- **gc_Start( )**
- **gc_Stop( )**
- **gc_UnListen( )**
- **gc_GetVoiceH( )**

- None of the call control related APIs are supported for clear channel trunks (DTI mode) (for example, **gc_MakeCall( )**, **gc_WaitCall( )**, **gc_AnswerCall( )**, etc.).
- For clear channel circuits, if a call control function is issued, an error message is generated indicating that the API is not supported. The error value EGC_UNSUPPORTED is the Global Call value returned when the **gc_ErrorInfo( )** function is used to retrieve the error code.

1.10.3 Documentation

The online bookshelf provided with System Release 6.1 for Linux contains information about all system release features including features for application development, configuration, administration, and diagnostics.

For more information about the Global Call API in general, see the following documents:

- *Global Call API Programming Guide*
- *Global Call API Library Reference*

For features specific to SS7 technology, see:

- *Global Call SS7 Technology Guide*

The online bookshelf has not been updated for this feature, so the *Global Call SS7 Technology Guide* does not currently indicate that SS7 boards are supported in a non SS7 signaling environment.

1.11 Additional Voice Channels on Clear Channel Media Loads

With the Service Update, customers using Clear Channel signaling now have basic voice support on the 31st channel. The new Clear Channel media loads maximize density of voice resources by fully utilizing the 31st channel on each trunk for DMV1200BTEP and DMV1200BTEC. On the resource boards, DM/V2400A-cPCI, and DMV4800BC, new media loads with additional voice channels can be used as resources for PSTN boards in the systems that are configured for Clear Channel.

1.11.1 Feature Description

Media loads are pre-defined sets of features supported by DM3 boards. A media load consists of a configuration file set (PCD, FCD, and CONFIG files) and associated firmware loads that are downloaded to each board. In most cases, the PCD/FCD/CONFIG file names indicate the associated media load and protocol. See the *DM3 Configuration Guide* for more information about media loads and configuration file sets.

The new media loads are listed in the following table:

Board	Media Load	PCD File Name	Basic Voice
DMV1200BTEP	1	gml1_qsb_4_e1cc*	124
DMV1200BTEC	1	gml1_cpciqsb_4_e1cc*	124
DM/V2400A-cPCI	1	ml1_cpcires_cc.pcd	248
DMV4800BC	1	ml1_cpciresb_cc.pcd	496
* These media loads are created using the trunk configuration utility. This new configuration can only run with the ML1; none of the other available loads for the boards are allowed. Also, the ML1 media loads support basic voice only; FSK is not supported.			

Notes:1. On the DMV1200BTEP and DMV1200BTEC boards, all spans must run clear channel; i.e., only clear channel is available during trunk configuration. Mixing of PDK/ISDN/Clear Channel is not permitted.

2. The resource boards media loads are meant to run with PSTN boards running Clear Channel.

3. On DMV2400A-cPCI, transaction record is not supported at full density; if transaction record is required, only 128 voice channels on the board can run (transaction record), while all other channels must remain idle due to timeslot limitations.

4. On DMV4800BC, transaction record is not supported at all.

5. On the DMV1200BTEP and DMV1200BTEC, transaction record can be run at full density.

1.11.2 Supported Boards

- Intel NetStructure® DMV1200BTEP Combined Media Board
- Intel NetStructure® DMV1200BTEC Combined Media Board
- Intel NetStructure® DM/V2400A Combined Media Board
- Intel NetStructure® DMV4800BC Combined Media Board

1.11.3 Documentation

The online bookshelf provided with System Release 6.1 for Linux contains information about all system release features including features for application development, configuration, administration, and diagnostics.

For detailed information about the supported boards, see the following document:

- *DM3 Configuration Guide*

1.12 Media Channel Reset Capability (Stuck Channel Recovery)

With the Service Update, whenever a media channel gets into a “stuck” state, there is a way to recover that channel without having to redownload the board or restart the application.)

Note: A stuck channel is defined as a failure where the host application is unable to recover the channel and no further media operations are possible on that channel until the board is redownloaded or (in some cases) the application is restarted.

1.12.1 Feature Description

In high-density applications, media channels sometimes become stuck and no further processing can take place until the board is redownloaded or (in some cases) until the application is restarted.

This feature provides new API functions in the voice library and in the continuous speech processing (CSP) library that enable the application to recover from the stuck channel and return it to an idle and usable state.

Note: Not all stuck channels are recoverable. Also, not all errors are stuck channel errors. See Section 1.12.2, “Restrictions and Limitations”, on page 49 for more information.

Supported Boards

All media span boards support this media channel reset feature, namely DM/V, DM/V-A, DM/V-B, and DM/IP boards.

New APIs

The two new API functions are as follows:

- **dx_resetch( )** - Call this API to recover the media channel when the channel is stuck and in a recoverable state. If the channel is recovered, a TDX_RESET event is generated to the application, which enables the application to reuse the channel for more media functions. If the channel is not in a recoverable state, a TDX_RESETERR event is sent back to the application indicating that the specific channel is not recoverable.
- **ec_resetch( )** - Call this API to recover the CSP channel when the channel is stuck and in a recoverable state. If the channel is recovered, TDX_RESET and TEC_RESET events are generated to the application, which enables the application to reuse the channel for more media functions. If the channel is not in a recoverable state, TDX_RESETERR and TEC_RESETERR events are sent back to the

application indicating that the specific channel is not recoverable. Note that the **ec_resetch( )** function resets both the voice and the CSP channels.

Function reference information is provided next.

dx_resetch( )

reset a channel that is hung

Name: dx_resetch (chdev, mode)

Inputs: int chdev
- valid channel device handle

int mode
- mode of operation

Returns: 0 if success
-1 if failure

Includes: srllib.h
dxxxlib.h

Category: I/O

Mode: asynchronous or synchronous

Platform: DM3

Description

The **dx_resetch( )** function recovers a channel that is "stuck" (busy or hung) and in a recoverable state, and brings it to an idle and usable state. This function blocks all other functions from operating on the channel until the function completes.

Parameter	Description
chdev	Specifies the valid device handle obtained when the channel was opened using **dx_open( )**
mode	Specifies the mode of operation: • EV_ASYNC – asynchronous mode. The calling thread returns immediately so it can process media functionality on other channels. • EV_SYNC – synchronous mode. The calling thread waits until the channel is recovered or discovers that the channel is not in a recoverable state.

In synchronous mode, 0 is returned if the function completes successfully, and -1 is returned in case of error.

In asynchronous mode, the TDX_RESET event is generated to indicate that the channel was recovered and is in an idle and usable state. The TDX_RESETERR event is generated to indicate that the channel is not recoverable. Issuing any other media calls on this channel will result in an error.

Cautions

- The **dx_resetch( )** function is intended for use on channels that are stuck and not responding. Do **not** use it in place of **dx_stopch( )**. Use **dx_resetch( )** only if you do not receive an event within 30 seconds of when it's expected. Overuse of this function creates unnecessary overhead and may affect system performance.

Errors

If the function returns -1, use the Standard Runtime Library (SRL) Standard Attribute function **ATDV_LASTERR( )** to obtain the error code or use **ATDV_ERRMSGP( )** to obtain a descriptive error message. One of the following error codes may be returned:

EDX_BADPARM
: Invalid parameter

EDX_FWERROR
: Firmware error

EDX_NOERROR
: No error

Example

```
#include <srllib.h>
#include <dxxxlib.h>

main()
{
   int chdev, srlmode;
   /* Set SRL to run in polled mode. */
   srlmode = SR_POLLMODE;

   if (sr_setparm(SRL_DEVICE, SR_MODEID, (void *)&srlmode) == -1) {
   /* process error */
   }

   /* Open the channel using dx_open( ). Get channel device descriptor in
   * chdev.
   */

   if ((chdev = dx_open("dxxxB1C1",NULL)) == -1) {
   /* process error */
   }

   /* continue processing */
   . .
   /* Force the channel to idle state. The I/O function that the channel
    * is executing will be terminated, and control passed to the handler
    * function previously enabled, using sr_enbhdlr(), for the
    * termination event corresponding to that I/O function.
    * In asynchronous mode, dx_stopch() returns immediately,
    * without waiting for the channel to go idle.
    */

   if ( dx_stopch(chdev, EV_ASYNC) == -1) {
   /* process error */
   }

   /* Wait for dx_stopch() to stop the channel and return the termination event
    * for the present media function.
    */

   /* After waiting for 30 secs if the termination event is not returned, issue a
    * dx_resetch() to reset the channel.
    */

   if (dx_resetch(chdev, EV_ASYNC) <0 )
   {
      /*process error */
```

```
    }

    /* Wait for TDX_RESET or TDX_RESETERR events */

}
```

See Also

- **ec_resetch**( ) in the *Continuous Speech Processing API Library Reference*

ec_resetch( )

reset a channel that is hung

Name: ec_resetch (chdev, mode)

Inputs: int chdev
- valid channel device handle

int mode
- mode of operation

Returns: 0 if success
-1 if failure

Includes: srllib.h
eclib.h

Category: I/O

Mode: asynchronous or synchronous

Platform: DM3

Description

The **ec_resetch( )** function recovers a channel that is "stuck" (busy or hung) and in a recoverable state, and brings it to an idle and usable state. This function blocks all other functions from operating on the channel until the function completes. This function recovers both the CSP channel and the voice channel.

Parameter	Description
chdev	Specifies the valid device handle obtained when the channel was opened using **dx_open( )**
mode	Specifies the mode of operation: • EV_ASYNC – asynchronous mode. The calling thread returns immediately so it can process media functionality on other channels. • EV_SYNC – synchronous mode. The calling thread waits until the channel is recovered or discovers that the channel is not in a recoverable state.

In synchronous mode, 0 is returned if the function completes successfully, and -1 is returned in case of error.

In asynchronous mode, the TDX_RESET and the TEC_RESET events are generated to indicate that the channel was recovered and is in an idle and usable state. The TDX_RESETERR and the TEC_RESETERR events are generated to indicate that the channel is not recoverable. Issuing any other media calls on this channel will result in an error.

Cautions

- The **ec_resetch( )** function is intended for use on channels that are stuck and not responding. Do **not** use it in place of **ec_stopch( )**. Use **ec_resetch( )** only if you do not receive an event within 30 seconds of when it's expected. Overuse of this function creates unnecessary overhead and may affect system performance.

Errors

If the function returns -1, use the Standard Runtime Library (SRL) Standard Attribute function **ATDV_LASTERR( )** to obtain the error code or use **ATDV_ERRMSGP( )** to obtain a descriptive error message. One of the following error codes may be returned:

EDX_BADPARM
: Invalid parameter

EDX_FWERROR
: Firmware error

EDX_NOERROR
: No error

Example

```
#include <stdio.h>
#include <srllib.h>
#include <dxxxlib.h>
#include <eclib.h>
#include <errno.h> /* include in Linux applications only; exclude in Windows */

main()
{
   int chdev, srlmode;
   /* Set SRL to run in polled mode. */
   srlmode = SR_POLLMODE;
   if (sr_setparm(SRL_DEVICE, SR_MODEID, (void *)&srlmode) == -1) {
   /* process error */
   }

   /* Open the channel using dx_open( ). Get channel device descriptor
   * in chdev.
   */
   if ((chdev = dx_open("dxxxB1C1",0)) == -1) {
   /* process error */
   }
   /* continue processing */
   . .
   /* Force the channel to idle state. The I/O function that the channel
   * is executing will be terminated, and control passed to the handler
   * function previously enabled, using sr_enbhdlr(), for the
   * termination event corresponding to that I/O function.
   * In the asynchronous mode, ec_stopch() returns immediately,
   * without waiting for the channel to go idle.
   */
   if (ec_stopch(chdev, FULLDUPLEX, EV_ASYNC) == -1) {
   /* process error */
   }

   /* Wait for the termination events (TEC_STREAM and/or TDX_PLAY) */

   /* After waiting for 30 secs, if the channel is still in a busy state,
    * issue ec_resetch() to reset both the CSP channel and the voice channel.
    * When issued in asynchronous mode, it will return both (TEC_RESET/TEC_RESETERR)
    * and (TDX_RESET/TDX_RESETERR) events.
    */

   if (ec_resetch(chdev, EV_ASYNC)  == -1 ) {
      /* process error */
   }
```

```
    /* Wait for TEC_RESET/TEC_RESETERR and TDX_RESET/TDX_RESETERR */
}
```

See Also

- **dx_resetch( )** in the *Voice API Library Reference*

Implementation Guidelines

The following guidelines apply when implementing the media channel reset capability using the voice API:

- It is recommended that you issue the function in asynchronous mode for more efficient processing. In synchronous mode, the calling thread is blocked until the function completes, which may take up to a minute in worst-case scenarios.
- The **dx_resetch( )** function is intended for use on channels that are stuck and not responding. Do **not** use it in place of **dx_stopch( )**. Use **dx_resetch( )** only if you do not receive an event within 30 seconds of when it's expected. Overuse of this function creates unnecessary overhead and may affect system performance.
- If you call **dx_resetch( )** immediately following **dx_stopch( )** without waiting at least 30 seconds for **dx_stopch( )** to complete, you will not receive events, such as TDX_PLAY and TDX_RECORD, even if the stop operation is successful and the channel was not stuck. Instead, you will only receive the TDX_RESET event if the channel recovery is successful or the TDX_RESETERR event if the channel is not recoverable.
- If you call **dx_resetch( )** without first using **dx_stopch( )** to stop the channel, the voice library will internally call **dx_stopch( )** and wait 30 seconds for it to complete. If the internal stop channel is successful, you will receive the TDX_RESET event only. If the internal stop channel is unsuccessful, the voice library will then call **dx_resetch( )**. Once a reset is attempted, you will receive the TDX_RESET event if the channel recovery is successful or the TDX_RESETERR event if the channel is not recoverable.
- Unrecoverable channels are written to a log file in the DebugAngel tool or the Runtime Trace Facility (RTF) tool. See the *Diagnostics Guide* for more information on these tools.

The following guidelines apply when implementing the media channel reset capability using the CSP API:

- The guidelines described for **dx_resetch( )** and **dx_stopch( )** apply to the **ec_resetch( )** and **ec_stopch( )** functions in the CSP API.
- For CSP applications, it is recommended that you use **ec_resetch( )** since this function resets both the voice and the CSP channels. The **dx_resetch( )** function resets the voice channels only.

1.12.2 Restrictions and Limitations

The following restrictions and limitations apply to the media channel reset feature:

- This feature only addresses scenarios where the firmware and the host library have lost synchronization or an event has not been propagated. DSP crashes, catastrophic firmware failures (killtasks), or unsynchronized firmware state machines are **not** recoverable without redownload of the board.

- This feature only addresses channels that become stuck while performing play and record, tone generation, or FSK operations. It also addresses channels that become stuck during CSP play or record operations.
- This feature does **not** address reset of IP media channels on DM/IP boards. It only addresses the reset of voice channels on DM/IP boards.
- The reset may not succeed if CPU utilization on the host system is close to 100 percent. It is recommended that the CPU usage be at a reasonable level (less than 70 percent) before you attempt a channel reset.

1.12.3 Documentation

The online bookshelf provided with System Release 6.1 for Linux contains information about all system release features including features for application development, configuration, administration, and diagnostics.

For more information about the Voice API, see the following documents:

- *Voice API Programming Guide*
- *Voice API Library Reference*

For more information about the CSP API, see the following documents:

- *Continuous Speech Processing API Programming Guide*
- *Continuous Speech Processing API Library Reference*

1.13 Additional Supported Operating System Distributions (OSDs)

With the Service Update, the following Linux Operating System Distributions (OSDs) are now supported:

- Red Hat* Enterprise Linux Version 4.0 Update 4 (Advanced Server, Enterprise Server, and Workstation)
- Red Hat* Enterprise Linux Version 4.0 Update 3 (Advanced Server, Enterprise Server, and Workstation)
- Red Hat* Enterprise Linux Version 4.0 Update 2 (Advanced Server, Enterprise Server, and Workstation)
- Red Hat* Enterprise Linux Version 4.0 Update 1 (Advanced Server, Enterprise Server, and Workstation)
- Red Hat* Enterprise Linux Version 3.0 Update 8 (Advanced Server, Enterprise Server, and Workstation)
- Red Hat* Enterprise Linux Version 3.0 Update 7 (Advanced Server, Enterprise Server, and Workstation)
- Red Flag* DC Server 5.0
- SUSE* Linux Enterprise Server 9 SP3

- SUSE* Linux Enterprise Server 9 SP2
- SUSE* Linux Enterprise Server 9

1.14 New Diagnostics Management Console

The Service Update introduces the Diagnostics Management Console (DMC) version 1.0. This GUI tool provides a means of quickly launching diagnostic utilities and viewing various log files created with those utilities.

The DMC:

- provides a single portal for launching diagnostic tools:
  - AppMon
  - Castrace
  - Isdntrace
  - Dlgsnapshot
  - Dm3post
  - Debugangel
  - Getver
  - its_sysinfo
  - Pdktrace
  - RTF Manager
- supports local and remote execution of tools. Diagnostic tools are launched remotely via the standard remote control methods provided with the operating system, such as SSH or Remote Desktop.
- lists the diagnostic logs available both locally and remotely for viewing.
- launches appropriate viewers for displaying logged data.

For more information, refer to the *Diagnostics Guide*. The DMC also has online help.

1.15 More Configurations for Optional Use of Sharing of Timeslot (SOT) Algorithm

With the Service Update, the ability to disable the Sharing of Timeslot (SOT) algorithm has been extended to support additional configurations:

- *ml10_dsa_ts16.pcd*
- *ml1b_dsa_ts16.pcd*
- *ml1b_qsa_ts16.pcd*
- *ml2_qsa_ts16.pcd*
- *ml5bc_dsa_ts16.pcd*

- *ml5bc_dsa_net5.pcd*

Note: For the table of supported boards and media loads, go to the website at: *http://resource.intel.com/telecom/support/tnotes/tnbyos/2000/tn104.htm*

For more details on the original configurations, refer to Section 1.29, “Optional Use of Sharing of Timeslot (SOT) Algorithm”, on page 63.

Feature Limitations, Caveats, or Restrictions

The user must modify the configuration file to add this functionality.

1.16 Ability to Send and Receive DPNSS End to End Messages

With the Service Update, the user has the ability to send and receive the entire raw Digital Private Network Signalling System (DPNSS) end to end message (EEM) using API control on the DM3 family of boards. A generic mechanism enables the user to add DPNSS supplementary services (like Single/Dual channel transfer services, Call Diversion, and Call Waiting) without needing outside support for those services first. This feature is only supported on ISDN DPNSS loads.

1.16.1 Feature Description

This feature enables the application to

- Enable GCEV_EXTENSION through **gc_SetConfigData( )** (for enabling the event).
- Send raw DPNSS EEM through **gc_SndMsg( )** with a new message type (for sending the event).
- Receive raw DPNSS EEM through GCEV_EXTENSION event on DM3 boards (for receiving the event).

The user has the ability to send and receive raw EEM frames. The user can extract the content of the EEM message and take the appropriate action when he/she receives any of the messages. The API is allowed in any intermediate call state. A majority of DPNSS supplementary services can be supported and the user does not need to request outside support for every new service that is being planned for the future.

EEM frames are of two types:

- EEM(I) - an end to end message (incomplete)
- EEM(C) - an end to end message (complete)

An EEM(C) is typically used, but if the size of the message exceeds 45 bytes in length, it can be split up into multiple EEM(I) messages, with a final piece of the message completed by an EEM(C).

Note: The application tracks the receipt of the various EEM(I) frames and reassembles them together to form the entire final EEM(C) message.

New Parameters

The feature is implemented using the Global Call **gc_SetConfigData( )** function and the GCTGT_CCLIB_CHAN parameter set. New extension event IDs define the receive raw DPNSS EEM through the GCEV_EXTENSION event. This unsolicited event can be enabled or disabled through **gc_SetConfigData( )**. The ISDN *dm3cc_param.h* header file is updated with the following:

- New extension ID: DM3CC_EXT_EVT_RAWEEM.
- New bit mask: EXTENSIONEVT_RAWEEM - Use to enable or disable the GCEV_EXTENSION event for DPNSS Raw EEM.
- New set ID: CCSET_RAWEEM.
- New PARMID: CCPARM_RAWEEM_DATA.

The ISDN *isdndef.h* header file is updated with SndMsg_RawEEM for the application to send raw EEM through **gc_SndMsg( )**.

Generated Events

GCEV_EXTENSION

The **gc_SetConfigData( )** function is issued to enable this functionality and the following notification event is generated for the application:

EXTENSIONEVT_RAWEEM
: Use to enable or disable the GCEV_EXTENSION event for DPNSS raw EEM.

The **gc_SndMsg( )** function is issued and the following notification events may be generated for the application:

CCSET_RAWEEM
: Receives raw EEM.

GCEV_TASKFAIL
: Indicates failure, for example, in case the information element (IE) has state change information in that the raw data contains an invalid IE or the raw data is 45 bytes.

Error Codes

The following success code is generated by the **gc_SetConfigData( )** function:

GC_SUCCESS
: Success. The signal type change has been implemented.

The following error code is generated by the **gc_SndMsg( )** function:

GCEV_TASKFAIL
Task failed. The FW will return Std_MsgError if the call state is not transferring because there is an invalid IE (call state changing), or the raw data is 45 bytes.

1.16.2 Enabling/Disabling GCEV_Extension Event

For the **gc_SetConfigData( )** function, the bit mask (EXTENSIONEVT_RAWEEM) is saved for use later during GCEV_EXTENSION event generation. The **gc_SetConfigData( )** is set on channel basis and has the target type set as GCTGT_CCLIB_CHAN. The general procedure is as follows:

Call the **gc_SetConfigData( )** function to implement the change.
gc_SetConfigData(GCTGT_CCLIB_CHAN.EXTENSIONEVT_RAWEEM)

1.16.3 Successfully Sending and Receiving Raw DPNSS EEM

The gc_SndMsg(MsgType, GC_IE_BLK) is used to send the raw DPNSS EEM like other DPNSS Supplementary Services (for example, Intrusion(SndMsg_Intrude), Diversion(SndMsg_Divert), NSI (SndMsg_NSI), etc.). The general procedures are as follows:

- To send an End to End Complete message, call the **gc_SndMsg( )** function with the msg_type parameter set to SndMsg_RawEEM. The first byte of the data portion (i.e., ie_Blk.data[0]) must contain 0x22 to indicate that it is an EEM(C) message. To receive the message, enable the GCEV_EXTENSION event.
- To send an End to End Incomplete message, call the **gc_SndMsg( )** function, with the msg_type parameter set to SndMsg_RawEEM. The first byte of the data portion (i.e., ie_Blk.data[0]) must contain 0x23 to indicate that it is an EEM(I) message. To receive the message, enable the GCEV_EXTENSION event.

The message is successfully received if no GCEV_TASKFAIL event is received at the user application.

Note: The first byte in the GC_IE_BLK is the spec defined Message ID for an EEM(I) or EEM(C) message.

Note: Certain supplementary information strings that may affect the firmware call state are not allowed in the raw EEM payload. Specifically not allowed are the HOLD-REQ string or 60B, and the RECON string or 61. If either of these strings is present, the application will receive a GCEV_TASKFAIL event.

Note: The total length of the raw EEM payload allowed is 45 bytes: 1 byte specifies the EEM type, which is EEM(C) or EEM(I), and 44 bytes is allowed for supplementary information strings encoded using the Backus Naur format and conforming to the DPNSS standard BTNR 188.

1.16.4 Sample Code

The following are samples of codes for sending raw EEM and receiving raw EEM.

To Send Raw EEM

```
int send_message(CRN crn)
{

 int      gc_err;          /* GlobalCall Error Code */
    int      cclibid;         /* Call Control library ID */
    long     cclib_err;       /* Call Control Error Code */
    char     *msg;            /* Error Message */
    LINEDEV  ldev;            /* Line device */
 char      str[MAX_STRING_SIZE];

 GC_IE_BLK gcIEBlk;
 IE_BLK ie_Blk;

 memset((unsigned char *)&ie_Blk, 0, sizeof(IE_BLK));

 gcIEBlk.gclib = NULL;
 gcIEBlk.cclib = &ie_Blk;
 ie_Blk.length =  7;       //length of the raw DPNSS EEM data
  /* EEM(C) = 0x22, EEM(I) = 0x23 */
 ie_Blk.data[0] = 0x22;   // raw DPNSS EEM data
 ie_Blk.data[1] = '*';    // raw DPNSS EEM data
 ie_Blk.data[2] = '1';    // raw DPNSS EEM data
 ie_Blk.data[3] = '1';    // raw DPNSS EEM data
 ie_Blk.data[4] = '0';    // raw DPNSS EEM data
 ie_Blk.data[5] = 'B';    // raw DPNSS EEM data
 ie_Blk.data[6] = '#';    // raw DPNSS EEM data

      if(gc_CRN2LineDev(crn, &ldev) != GC_SUCCESS) {
        gc_ErrorValue(&gc_err, &cclibid, &cclib_err);
        gc_ResultMsg(cclibid, cclib_err, &msg);
  sprintf(str, "Error on Device handle : 0x%lx ",ldev);
  printandlog(0, GC_APICALL, NULL, str, 0);
        return(cclib_err);
    }

    if(gc_SndMsg(ldev, crn, SndMsg_RawEEM, &gcIEBlk) != GC_SUCCESS) {
        gc_ErrorValue(&gc_err, &cclibid, &cclib_err);
        gc_ResultMsg(cclibid, cclib_err, &msg);
        sprintf(str, "Error on Device handle : 0x%lx ",ldev);
  printandlog(0, GC_APICALL, NULL, str, 0);
        return(cclib_err);
    }
    return 0 ;
}
```

To Enable the GCEV_EXTENSION Event to Receive Raw EEM Events

```
int EnableRawEEMInformation(int DeviceHdl)
{
 GC_PARM_BLKP pParmBlock = NULL;
 long requestID;
 char      str[MAX_STRING_SIZE];

 int iRetCode = gc_util_insert_parm_val(&pParmBlock, CCSET_EXTENSIONEVT_MSK,
  GCACT_ADDMSK, sizeof(long), EXTENSIONEVT_RAWEEM);
```

```
 int rc = gc_SetConfigData(GCTGT_CCLIB_CHAN, DeviceHdl, pParmBlock,0,
  GCUPDATE_IMMEDIATE, &requestID, EV_ASYNC);

 if(rc != GC_SUCCESS) {
  sprintf(str, "failed to set evt mask");
  printandlog(0, GC_APICALL, NULL, str, 0);
  return GC_ERROR;
 } else {
  sprintf(str, "gc_SetConfigData() called - Raw EEM event  reception enabled");
  printandlog(0, GC_APICALL, NULL, str, 0);
 }

 gc_util_delete_parm_blk(pParmBlock);

 return 0;
}
```

To Receive Raw EEM and Extract Raw DPNSS Data

```
void process_event(void)
{ ....
  ....
  ....
 switch (evttype)
 {
  case GCEV_EXTENSION:
   ExtractDPNSSInfo(pline, &metaevent);
   break;

 }
}

void ExtractDPNSSInfo(struct channel *pline,METAEVENT *metaeventp)
{
 GC_PARM_BLKP gcParmBlkp = NULL;
 GC_PARM_DATAP t_gcParmDatap = NULL;
 EXTENSIONEVTBLK *ext_evtblkp = NULL;

 GC_IE_BLK * t_gcIEBlk = NULL;
 IE_BLK * ie_blk = NULL;
 char rawData[100];
 char str[MAX_STRING_SIZE];
 int i=0;

 ext_evtblkp = (EXTENSIONEVTBLK *)metaeventp->extevtdatap;
 gcParmBlkp = &ext_evtblkp->parmblk;

 sprintf(str, "Received GCEV_EXTENSION event with ExtID = 0x%x",ext_evtblkp->ext_id);
 printandlog(0, GC_APICALL, NULL, str, 0);
 while (t_gcParmDatap = gc_util_next_parm(gcParmBlkp, t_gcParmDatap))
 {
 switch (t_gcParmDatap->set_ID)
  {
   case CCSET_RAWEEM:
    switch(t_gcParmDatap->parm_ID)
    {
     case CCPARM_RAWEEM_DATA:
         t_gcIEBlk =  (GC_IE_BLK *)t_gcParmDatap->value_buf;
      ie_blk =  t_gcIEBlk -> cclib;
      memcpy(rawData, ie_blk->data,ie_blk->length);

      sprintf(str, "RAWEEM_DATA : length = %d\n", ie_blk->length);
      printandlog(0, GC_APICALL, NULL, str, 0);
      memset(str, 0, MAX_STRING_SIZE);
```

```
        for (i=0; i < ie_blk->length; i++)
        {
         if((i!=0) && (isascii(rawData[i]))) {
          printf(str, "%c ", rawData[i]);

          fprintf(port[0].log_fp, "%c ", rawData[i]);
         }

         else {
          printf(str, "%02X ", rawData[i]);
          fprintf(port[0].log_fp, "%02X ", rawData[i]);
         }
        }
        printf("\n");
        fprintf(port[0].log_fp, "\n ");

       break;
       default:
        sprintf(str, "Unknown PARM ID");
        printandlog(0, GC_APICALL, NULL, str, 0);
       break;
      }
     break;
    default:
     sprintf(str, "Unknown SET ID");
     printandlog(0, GC_APICALL, NULL, str, 0);
    break;
   }
  }

}
```

1.16.5 Documentation

The online bookshelf provided with the system release contains information about all system release features including features for application development, configuration, administration, and diagnostics.

For more information about the Global Call API, see the following documents:

- *Global Call API Library Reference*
- *Global Call API Programming Guide*

For features specific to ISDN technology, see:

- *Global Call ISDN Technology Guide*

Note: The online bookshelf has not been updated for this feature, so the *Global Call ISDN Technology Guide* does not currently include information about sending and receiving raw DPNSS end to end messages.

1.17 Time Stamp for Tone-On/Off Events

With the Service Update, a new time stamp has been added to the existing DE_TONEON and DE_TONEOFF events. A new TN_TIMESTAMP structure has been added to the device header file *dxxxlib.h*. This time stamp is used to associate, or group, certain tones

in order to detect a particular country tone made up of two or more defined tone templates.

1.17.1 Feature Description

To test the various tones from various countries, the Tone-On/Off Call Status Transition (CST) event data have been modified to add a time stamp structure to the end of the TN_INFO structure. The CST event data are obtained by calling **sr_getdatalen( )** and **sr_getevtdatap( )**. A new structure, TN_TIMESTAMP, is in the device header file, *dxxxlib.h*. If the event is for Tone-On, then the time stamp represents the tone-on time, and if the event is for Tone-Off, then it represents the tone-off time.

The Tone-On/Tone-Off messages are extended to add the “start time” and the “stop time,” respectively. These time stamps are used by the customer application to calculate the Tone-On/Tone-Off duration (cadence).

1.17.2 Supported Boards

The following board is supported:

- Intel NetStructure® DM/V2400A

1.17.3 Structure

TN_TIMESTAMP is as follows:

```
// Tone ON/OFF time stamp
typedef struct {

    unsigned long  tn_TimeStamp;   /* Time stamp for tone on/off event. The time stamp is in
                                      milliseconds from when the firmware was downloaded on the
                                      board. There is no co-relation to the system time. It wraps
                                      around every ~149 hours. */
} TN_TIMESTAMP;
```

Scenario

When a Tone-On CST event is received, the application gets the CST event data with the **sr_getdatalen( )** and **sr_getevtdatap( )** functions, as usual. The application then applies the TN_TIMESTAMP structure to the event data and obtains the time stamp of the tone-on event or tone-off event. The TN_TIMESTAMP structure is appended to the end of the TN_INFO structure. The CST event data comprises the DX_CST, TN_INFO, and TN_TIMESTAMP structures.

Sample

The following is an example for Tone-On. Tone-Off is done the same way.

```
DX_CST *datap;
TN_INFO *tonep;
TN_TIMESTAMP *tsp;
long timestamp; // time stamp in ms units

switch(sr_getevttype(ehandle))
{
     case TDX_CST:
         datap = (DX_CST *) sr_getevtdatap(ehandle);

         if (datap->cst_event == DE_TONEON)
         {
             tonep = (TN_INFO*)(datap+1); // tone structure starts at end of CST structure
             tsp = (TN_TIMESTAMP*)(tonep+1); // time stamp structure starts at end of
                                                                    TN_INFO structure.
             timestamp = tsp->tn_TimeStamp; // get the time stamp
}

break;
.
.
```

1.17.4 Documentation

The online bookshelf provided contains information about all system release features including features for application development, configuration, administration, and diagnostics.

For more information about the Standard Runtime Library and Voice APIs, see the following documents:

- *Standard Runtime Library API Library Reference*
- *Standard Runtime Library API Programming Guide*
- *Voice API Programming Guide*
- *Voice API Library Reference*

Note: The online bookshelf has not been updated for this feature, so the manuals listed above do not contain information relating to this feature.

1.18 OA&M Error Cleanup

With the Service Update, OA&M (operations, administration, and maintenance) errors continue to be logged by the Runtime Trace Facility (RTF), as well as /usr/var/syslog. These errors have been made clearer, more concise, and more consistent.

1.19 New Runtime Trace Facility (RTF) Manager

The Service Update introduces the RTF Manager, a new GUI for the Runtime Trace Facility (RTF) diagnostic tool. RTF Manager allows users to easily configure logging and tracing levels. Previously, users had to manually edit the RTF configuration file.

For more information about the RTF Manager, refer to the *Diagnostics Guide*.

1.20 New Application Monitor

The Service Update introduces the Application Monitor diagnostic tool. Users can set up the Application Monitor to monitor one or more applications and if a problem occurs, it will launch the its_sysinfo tool to collect data that can help in diagnosing the problem.

For more information about the Application Monitor, refer to the *Diagnostics Guide*.

1.21 Improved Tracing and Error Reporting

With the Service Update, the errors and warnings that are propagated up the software stack have been made clearer, more concise, and more consistent.

1.22 New Media Load for DMV600BTEC Boards

The Service Update provides a new media load, ML FN, for the Intel NetStructure® DMV600BTEC Combined Media Board. This new media load provides a higher density of fax resources than other media loads for this board. ML FN provides 60 fax channels and 60 network interfaces (with tone resources) on a single board.

1.22.1 Feature Description

Predefined sets of features for Intel NetStructure® boards are provided in media loads. A media load consists of a configuration file set (PCD, FCD, and CONFIG files) and the associated firmware that is downloaded to the board. See the *DM3 Configuration Guide* for more information about media loads.

The features and channel densities provided by media load ML FN are as follows:

Features Supported	Voice	Fax	Network Interfaces
Channel Density	0	60	60

Note: The 2048-pixel and 2432-pixel formats are supported only on the first 32 channels. The last 28 channels support 1728-pixel width only. It is up to the application to keep track of which channels support the 2432-pixel width. Attempting to transmit 2048-pixel or 2432-pixel formats on the last 28 channels will result in a TFX_FAXERROR. Attempting to receive 2048-pixel or 2432-pixel formats on the last 28 channels will result in a truncated image if the rcvflag is set to 1728-pixel or a TFX_FAXERROR if rcvflag is not set to 1728-pixel. If reception of a 2432-pixel page width image is attempted on the last 28 channels, the received image will be scaled to 1728 width provided the workaround below is used. If this workaround is not used, the reception of a 2432 width image will fail.

Workaround: As part of the required receive fax API call, “fx_rcvfax”, the application

should also explicitly include the maximum receive width as follows: (h is the Fax handle)

First 32 channels of load:
fx_rcvfax(h , …, EV_ASYNC | DF_PHASEB | DF_PHASED | DF_2432MAX)

Last 28 channels of load:
fx_rcvfax(h , …, EV_ASYNC | DF_PHASEB | DF_PHASED | DF_1728MAX)

ML FN is supported for ISDN and PDK protocols, including mixed ISDN/CAS (i.e., ISDN on one trunk and CAS on the other trunk).

1.22.2 Configuring the Software

The new media load can be selected when using the Intel® Dialogic® Configuration Manager for Linux. The configuration procedure is described in detail in the *DM3 Configuration Guide*. This is done before the boards are started.

1.22.3 Documentation

The online bookshelf provided with System Release 6.1 for Linux contains information about all system release features including features for application development, configuration, administration, and diagnostics.

For detailed information about configuring DMV600BTEC boards, see the following document:

- *DM3 Configuration Guide*

Note: The online bookshelf has not been updated for this feature, so the *DM3 Configuration Guide* does not currently include information about media load ML FN.

1.23 Removal of IP Gateway R4 and IPML Gateway Demos

With the Service Update, the IP Gateway R4 and IPML Gateway demos have been removed from build and no longer supported in System Release 6.1 for Linux.

1.24 New Status Monitor (statusmon) Application

The Service Update introduces the Status Monitor (statusmon) application for Linux, which has previously been available only for Windows. Statusmon is a command line application with a terminal user interface (TUI). The application supports two execution modes:

- Monitoring call state.

- Monitoring bit information.

For more information about the statusmon application, refer to the *Diagnostics Guide*.

1.25 New Version of Runtime Trace Facility (RTF) Tool

The Service Update introduces a new version of the Runtime Trace Facility (RTF) tool. The RTF tool provides a mechanism for tracing the execution path of runtime libraries that are supported by System Release 6.1 for Linux. Support for the RTF tool is as follows:

- `Rtftool` command is used to stop/start the RTF tool's tracing capabilities.
- Provides centralized logging for key OA&M components (OAMSYSLOG) and IP libraries.
- Run the RTF tool in preservation mode. Preservation mode allows you to save specified RTF trace information into a separate, preserved log file while the RTF engine continues to output active trace information into the default log file. The RTF engine will not overwrite, delete or append to the preserved log file after it has been saved.
- Trace the Audio Conferencing API (DCB) library.

For more information about the RTF tool, refer to the *Diagnostics Guide*.

1.26 New Version of Get Version (Getver) Tool

The Service Update introduces a new version of the Get Version (getver) tool. Getver outputs version information for files that are part of the Intel Telecom software installation.

For more information about the getver tool, refer to the *Diagnostics Guide.*

1.27 New Version of its_sysinfo Tool

The Service Update introduces a new version of the Intel Telecom Subsystem Summary (its_sysinfo) tool. The its_sysinfo tool collects data from the system on which you execute it and provides you with information about the system environment: the operating system, computer architecture, Intel Dialogic System Release software, and operational logs. The new version of its_sysinfo supports a command line interface, provides more data-collecting features, and improves some existing data collecting features.

For Linux systems, the *its_sysinfo.htm* file now includes a Linux Package Info section at the beginning of the file. For example:

```
                    LinuxPackageInfo

Installed Dialogic RPM Information
```

```
DLGChdsi-6.2.0.0
Software for the Intel NetStructure(R) HDSI
Install Date: Tue 14 Nov 2006 07:36:50 PM EST
Build Host:   myhost
Build Date:  Thu 09 Nov 2006 02:15:31 PM EST

DLGCsdk-6.2.0.0
Intel(R) Dialogic(R) Software Development Kit
Install Date: Tue 14 Nov 2006 07:39:09 PM EST
Build Host:   myhost
Build Date:  Thu 09 Nov 2006 02:15:04 PM EST

...

Selections Made from Dialogic Install Menu

Intel Netstructure HDSI
Intel Netstructure DMIP
Global Call Protocols
Intel Dialogic Boards
...
```

For more information about the its_sysinfo tool, refer to the *Diagnostics Guide.*

1.28 Support for Open Source Drivers

With the Service Update, Open Source Drivers are now supported. The Open System Release project is the collaborative development effort for System Release 6.1 for Linux. By providing source code and development services, Intel encourages our customers to participate in the improvement and evolution of future system releases.

For Intel community development support, visit the Open System Release website at the following link:
http://www.opensystemrelease.com

1.29 Optional Use of Sharing of Timeslot (SOT) Algorithm

The Service Update now supports disabling the Sharing of Timeslots (SOT) algorithm for certain media loads. The SOT algorithm for DMV, DM/IP, and DM/V-A boards with network interfaces maximizes the efficiency of the internal timeslots used for external transmit reference, allowing a full 120 channel density for such features as continuous speech processing and transaction record. The SOT algorithm is enabled by default, regardless of whether continuous speech processing or transaction record functionality is needed. Its use places certain constraints on an application for performing listen/unlisten functions in a specific sequence.

Note: The SOT algorithm does not apply to the resource-only boards: DM/V-B, DM/V-A, and DMV/IP.

For increased flexibility in application design, it is now possible to disable the SOT algorithm by adding a new parameter, **QKERNEL_DISABLE_TIMESLOT_SHARING**, to the board's .config file, and then using fcdgen to generate an updated .fcd file.

For more detailed information about the SOT algorithm, guidelines for enabling or disabling the algorithm, and supported boards and media loads, see the technical note titled "Disabling the Sharing of Timeslot (SOT) Algorithm via DM3 config file change" on the Telecom Support Resources website at:
http://resource.intel.com/telecom/support/tnotes/tnbyos/2000/tn104.htm

1.30 Deprecation of Dlgcsnmpd and Orbacus Init Scripts

In an upcoming Service Update build, two of the start/stop scripts previously included in the release will be removed. In general, these scripts are used by the system at boot time to start/stop the Intel Dialogic services and are not generally accessed by users.

/etc/init.d/orbacus

This script is used to start or stop the CORBA-based framework currently used by the Intel Dialogic System Release software. In an upcoming Service Update build, the need for this CORBA implementation has been eliminated, rendering the script unnecessary. No replacement will offered for this interface.

/etc/init.d/dlgcsnmpd

This script is used to start or stop the Intel Dialogic SNMP support. The functionality of this script has already been moved into the main Intel Dialogic System Release start/stop script, /etc/init.d/ct_intel and in an upcoming Service Update build, the dlgcsnmpd script will be removed. A table mapping old dlgcsnmpd functionality to the updated ct_intel script is provided below so that users directly accessing the dlgcsnmpd script can modify their implementations to use the ct_intel script instead.

ACTION	OLD	NEW
start Intel Dialogic SNMP	/etc/init.d/dlgcsnmpd start	/etc/init.d/ct_intel snmpstart
stop Intel Dialogic SNMP	/etc/init.d/dlgcsnmpd stop	/etc/init.d/ct_intel snmpstop
restart Intel Dialogic SNMP	/etc/init.d/dlgcsnmpd restart	/etc/init.d/ct_intel snmprestart
check Intel Dialogic SNMP status	/etc/init.d/dlgcsnmpd status	/etc/init.d/ct_intel snmpstatus

The status returned by the 'ct_intel snmpstatus' command has been enhanced to show all of the three programs involved in the SNMP support. Previously, dlgcsnmpd only displayed the status of one of the three.

The elimination of these two scripts will result in a single point-of start/stop control for all of the Intel Dialogic System Release software and eliminate certain issues which occurred when one of the three services was stopped without the other two being stopped.

If you select the SNMP item from the install menu, then the SNMP support will be started at system boot or when dlstart is executed. If you do not want the SNMP support to be started:

1. Do not select “All” or “SNMP Component Manager” from the install menu.
2. Do not install the net-snmp RPMs on your system. Intel Dialogic SNMP support will not start if this dependency is not found.
3. Do not configure SNMP during the Intel Dialogic System Release configuration process.

1.31 Removal of Unused Header Files

With the Service Update, the following header files that were included in previous System Release 6.1 for Linux builds are being removed:

- ICTDm3Board.h
- ICTPmacBoard.h
- ICTPreConfigMgr.h
- ICTSprwBoard.h

These header files were inadvertently included in previous builds and are being removed since the files have no functional implementation. The removal has no functional impact on any runtime functionality that is supported for management applications built on the existing DASI implementation. If these files were included in any make rules for management applications, the associations to these header files need to be removed to avoid inadvertent compile-time errors.

1.32 Dynamic Detection of ETSI FSK Protocols

The Service Update adds a new mode for ETSI-compliant frequency shift keying (FSK) signaling in **dx_setparm( )** in the Voice API library. This mode is used in short message service (SMS) applications, and enables Intel NetStructure® boards to dynamically switch between ETSI Protocol 1 and ETSI Protocol 2 as needed to complete the SMS transmission.

This new functionality is supported on Intel NetStructure® DM/V-A and DM/V-B boards.

1.32.1 Feature Description

The Voice API library currently supports both ETSI Protocol 1 and ETSI Protocol 2 for FSK signaling in short message service (SMS) applications. Protocol 1 is primarily for SMS applications using GSM with mobile devices. Protocol 2 is primarily for SMS applications using PSTN or ISDN in fixed residential networks.

The ETSI Protocol 1 or ETSI Protocol 2 is specified on a channel basis using the channel seizure and mark length parameters in **dx_setparm( )**: DXCH_FSKCHSEIZURE and

DXCH_FSKMARKLENGTH. These parameters allow the protocol to be configured in a static way between a calling party and a called party. This static definition works in an environment where the protocol to be used in an SMS transmission is known.

In some environments, however, the protocol used by a called party (ETSI Protocol 1 or ETSI Protocol 2) is unknown to the telecom board. In these cases, the telecom board needs to dynamically detect the protocol used by the called party in each SMS session, adjust its configuration to match the protocol in use, and successfully complete the SMS transmission.

The following new mode for DXCH_FSKSTANDARD in **dx_setparm( )** is provided to enable the ETSI protocol auto-detect feature: DX_FSKSTDETSIAUTO.

The updated description for DXCH_FSKSTANDARD is as follows:

DXCH_FSKSTANDARD

Specifies the FSK protocol standard used for transmission and reception of FSK data. The protocol standard can be set to one of the following:

- DX_FSKSTDBELLCORE – Bellcore standard (default value)
- DX_FSKSTDETSI – ETSI standard
- DX_FSKSTDETSIAUTO – Auto-detect ETSI Protocol 1 or ETSI Protocol 2

If you specify DX_FSKSTDETSIAUTO, it is recommended that you call **dx_RxIottData( )** to let the firmware determine the specific values for the DXCH_FSKCHSEIZURE and DXCH_FSKMARKLENGTH. The values are set based on the results of the protocol negotiation with the called party.

Notes:1. The ETSI auto-detect mode (DX_FSKSTDETSIAUTO) applies only in cases where the short message service center (computer where Intel telecom boards reside) is the calling party; that is, the short message service center receives the Data Link Establish message (DLL_SMS_EST). For sessions where the short message service center is the called party (that is, the short message service center sends the DLL_SMS_EST message), the application is responsible for establishing the proper FSK standard, channel seizure, and mark length values. In this case, if the wrong parameters are set, the protocol will time out and it is up to the application to retry the transmission.

2. In ETSI auto-detect mode, the channel seizure and mark length values retrieved for a called party remain in effect for subsequent sessions on the same channel, unless modified using **dx_setparm( )**.

1.32.2 Using the DX_FSKSTDETSIAUTO Mode

The following steps briefly describe how to enable and use ETSI auto-detect mode in an SMS application:

1. After opening a channel device, call **dx_setparm( )** and specify the ETSI auto-detect mode, DX_FSKSTDETSIAUTO, for DXCH_FSKSTANDARD.
2. Call **dx_RxIottData( )** to receive FSK data on the specified channel. The firmware determines the appropriate values for channel seizure and mark length based on the results of the protocol negotiation with the called party.
3. After **dx_RxIottData( )** has completed, you can call **dx_getparm( )**, if desired, to retrieve the channel seizure and mark length values and determine whether the called party uses ETSI Protocol 1 or ETSI Protocol 2. Note that this step is not necessary to complete the current SMS session as the firmware will automatically detect the channel seizure and mark length values.

1.32.3 Documentation

For more information about sending and receiving FSK data, see the *Voice API Programming Guide*. For details about **dx_setparm( )** and other Voice API library functions, see the *Voice API Library Reference*. For details about the ETSI protocols, see the ETSI ES 201 912 specification.

1.33 dx_stopch( ) EV_NOSTOP Mode Support for DM3 Boards

With the Service Update, the **dx_stopch( )** function EV_NOSTOP mode can now be used with DM3 boards. The **dx_stopch( )** function forces termination of currently active I/O functions on a specified channel. It forces a channel in the busy state to become idle. When issued on a channel that is already idle and EV_NOSTOP is specified, the application receives a TDX_NOSTOP event to indicate that the **dx_stopch( )** function completed and that no STOP was needed (that is, the channel was already idle when **dx_stopch( )** was issued). Previously, no event was returned in this situation for DM3 boards.

1.33.1 Feature Description

EV_NOSTOP can be ORed with any of the other mode flags supported by **dx_stopch( )**, for example:

```
dx_stopch(chdev, EV_ASYNC|EV_NOSTOP)
```

dx_stopch( ) with EV_NOSTOP operates as follows:

- If issued on an idle channel, TDX_NOSTOP is returned.

- If issued on an active channel, the active channel is stopped and a media termination event (e.g., TDX_PLAY) is returned with termination reason TM_USRSTOP. (TDX_NOSTOP is not returned.)
- If issued in synchronous mode (EV_SYNC | EV_NOSTOP), no event is sent to application. **dx_stopch( )** will return only when the channel comes to idle state.

Note that it is possible to get both events (TDX_PLAY and TDX_NOSTOP) in a race condition, that is, if **dx_stopch( )** is issued for a channel that transitioned to idle just before the stop request, before the application received the TDX_PLAY event. In this case, the termination reason for the TDX_PLAY is **not** TM_USRSTOP.

1.33.2 Documentation

The online bookshelf provided with System Release 6.1 for Linux contains information about all system release features including features for application development, configuration, administration, and diagnostics.

For more information about the **dx_stopch( )** function and the Voice API, see the following documents:

- *Voice API Programming Guide*
- *Voice API Library Reference*

1.34 Supported Kernel and GCC Versions

With the Service Update, the newly supported OSDs (Red Hat* Enterprise Linux Version 4.0 and SUSE* Linux Enterprise Server 9) are based on the 2.6 kernel. The Service Update will support both 2.6 and 2.4 kernel based OSDs. Redundant Host and Peripheral Hot Swap will not be supported with the 2.6 kernel in the current Service Update.

All of the supported OSDs do not use the same version of GCC, so the System Release 6.1 for Linux Service Update CD includes RPMs for two different versions of GCC: 3.2 and 3.4. For more information, refer to the *Software Installation Guide*. The *Release Guide* contains the list of OSDs supported for the original release as well as additional details about system requirements.

1.35 New Media Loads for DMV1200BTEC Boards

The Service Update provides new media loads for the Intel NetStructure® DMV1200BTEC Combined Media Boards.

ML QSB-U2
: Provides increased density for conferencing (with echo cancellation and tone clamping), while also providing basic voice, FSK, transaction record, and fax.

ML 10B

Provides basic voice, FSK, transaction record, and conferencing (with echo cancellation and tone clamping), with channel densities designed for use with high density background music applications.

Background music is an application where two callers chat while music plays in the background. In essence, it is a 3-party conference with the third party being a voice resource playing music. Background music applications require a ratio of 2 network interfaces to 1 voice resource to 3 conference parties. The network interfaces on the DMV1200BTEC board can be used, thus providing a one-board solution.

Note: ML 10B is only supported for ISDN protocols.

1.35.1 Feature Description

Predefined sets of features for Intel NetStructure® boards are provided in media loads. A media load consists of a configuration file set (PCD, FCD, and CONFIG files) and the associated firmware that is downloaded to the board. See the *DM3 Configuration Guide* for more information about media loads.

Media Load QSB-U2

The features and channel densities provided by media load QSB-U2 are as follows:

Features Supported	Basic Voice (ML1), FSK, and Transaction Record	Fax	Rich Conferencing (ML9B), with Echo Cancellation and Tone Clamping
Channel Density	120	12	120

There are 120 total voice resources. Any combination of the voice features (basic voice, FSK, and transaction record) can be used up to a total of 120. There are no limitations in transaction record density supporting it at full density (120). In addition to the voice resources, 12 fax resources and 120 conferencing resources (with echo cancellation and tone clamping) can be used concurrently.

Conference size is limited to 20 parties without bridging. Conference bridging can be used to effectively expand a conference beyond the maximum size. Conference bridging consumes conferencing resources, reducing overall board conference density.

Media Load 10B

The features and channel densities provided by media load 10B are as follows:

Features Supported	Basic Voice (ML1), FSK, and Transaction Record	Rich Conferencing (ML9B), with Echo Cancellation or Tone Clamping
Channel Density	60	180

There are 60 total voice resources. Any combination of the voice features (basic voice, FSK, and transaction record) can be used up to a total of 60. There are no limitations in

transaction record density supporting it at full density (60). In addition to the voice resources, 180 conferencing resources (with echo cancellation and tone clamping) can be used concurrently.

Conference size is limited to 18 parties without bridging. Conference bridging can be used to effectively expand a conference beyond the maximum size. Conference bridging consumes conferencing resources, reducing overall board conference density. This is not an issue for background music applications, which use 3-party conferences.

Note: ML 10B can be used with any ISDN protocol supported by the DMV1200BTEC board, but it cannot be used with CAS and R2MF protocols.

1.35.2 Configuring the Software

The new media loads can be selected when using the Intel® Dialogic® Configuration Manager for Linux. The configuration procedure is described in detail in the *DM3 Configuration Guide*. This is done before the boards are started.

In addition, the following runtime parameter changes are recommended when playing music into a conference (e.g., when using ML 10B):

- When you add music to the conference, set its party attributes so that it uses transmit-only mode. That is, for the conference party that transmits music, enable the MSPA_MODEXMITONLY attribute in the MS_CDT data structure chan_attr field.
- The Conferencing AGC Noise Level Lower Threshold may have to be adjusted. This parameter is set at -40 dB by default and filters out any signals below this level. If the background music levels are low, the music may not be summed into the conference or it may come in and out. In this case, the AGC Noise Level Lower Threshold should be reduced by adding “setparm=0x3b1f, <value>” to the [0x3b] section of the CONFIG file and re-downloading the board. For further information about this parameter, refer to the discussion of CSUMS_AGC_low_threshold (AGC Noise Level Lower Threshold) in Section 1.43, “New Media Load for DMV4800BC Boards”, on page 75.

1.35.3 Documentation

The online bookshelf provided with System Release 6.1 for Linux contains information about all system release features including features for application development, configuration, administration, and diagnostics.

For detailed information about configuring DMV1200BTEC boards, see the following document:

- *DM3 Configuration Guide*

For information about the Audio Conferencing (DCB) API, see the following documents:

- *Audio Conferencing API Programming Guide*

- *Audio Conferencing API Library Reference*

Note: The online bookshelf has not been updated for this feature, so the *DM3 Configuration Guide* does not currently include information about media loads QSB-U2 and 10B.

1.36 GSM-AMR-NB Coder Support for IPT Boards

With the Service Update, Groupe Systeme Mobile (GSM) Adaptive Multi-Rate (AMR) Narrow Band (NB) coders are supported on Intel NetStructure® IPT boards. Any AMR-NB allowable rate of 4.75 to 12.2 kbps may be used. Full density of AMR coders (up to 480 channels) is supported.

Note: In order to support this feature, the IPT board must have a revision D0 Digital Signal Processor (DSP). Older revision boards do not support this feature. To determine if your board can use the AMR-NB coders, check the Configured Assembly (CA) label, which is located on the same side of the board as the four MAC address labels. (It is in close proximity to the front panel and parallel to it.) Your board must have one of the CA numbers listed in the following table to use the AMR-NB coders.

Board Model	CA Number
IPT6720C	C48123-001 or C48123-003
IPT4800C	C56297-003
IPT2400C	C56004-003
IPT1200C	C52890-001 or C52890-003

1.36.1 Feature Description

Coder information can be specified using the IP Media Library API.

1.36.1.1 Using the IP Media Library API to Specify the Coder

The **ipm_StartMedia( )** function in the IP Media Library API sets media properties and starts a media session. Coder information is provided in the IPM_MEDIA_INFO data structure. The IPM_MEDIA_INFO structure references IPM_MEDIA, which in turn references IPM_CODER_INFO for coder information.

The following new values are supported for the eCoderType field of the IPM_CODER_INFO structure:

- CODER_TYPE_AMRNB_4_75k - GSM-AMR-NB 4.75 kbps
- CODER_TYPE_AMRNB_5_15k - GSM-AMR-NB 5.15 kbps
- CODER_TYPE_AMRNB_5_9k - GSM-AMR-NB 5.9 kbps
- CODER_TYPE_AMRNB_6_7k - GSM-AMR-NB 6.7 kbps
- CODER_TYPE_AMRNB_7_4k - GSM-AMR-NB 7.4 kbps
- CODER_TYPE_AMRNB_7_95k - GSM-AMR-NB 7.95 kbps
- CODER_TYPE_AMRNB_10_2k - GSM-AMR-NB 10.2 kbps

- CODER_TYPE_AMRNB_12_2k - GSM-AMR-NB 12.2 kbps

  Note: These coder types are supported only on Intel NetStructure® IPT boards.

When using any of these coders:

- Coder frame size is fixed at 20 ms.
- Frames per packet (fpp) is:
  - - 1 or 2 (transmit)
  - - 1, 2, or 3 (receive)
- VAD must be disabled.

1.36.2 Release Documentation

The online bookshelf provided with System Release 6.1 for Linux contains information about all system release features including features for application development, configuration, administration, and diagnostics.

For more information about the IP Media Library API, see the following documents:

- *IP Media Library API Programming Guide*
- *IP Media Library API Library Reference*

1.37 Tcl/Tk No Longer Required for QScript Utilities Support

With the Service Update, an independent Tcl/Tk package no longer needs to be installed because it has been incorporated as a component of the System Release 6.1 for Linux build. Also, no symbolic links are required to simulate Tcl/Tk 8.3.

1.38 System Logging Integrated with Runtime Trace Facility (RTF)

With the Service Update, the System Logging facility has been integrated with Runtime Trace Facility (RTF). The environment DLG_TRACE_LEVEL is replaced by the RTF module OAMSYSLOG. The dlgsyslogger.log and oam.log files are no longer generated and have now been replaced by RTF log files. Refer to the *Diagnostics Guide* for more details on RTF logging.

1.39 Support for Multiple SS7 Boards in the Same System

With the Service Update, Global Call SS7 supports multiple SS7 boards in the same system. Previous versions of Global Call SS7 supported operation with a single SS7

board in the host system. Starting with the Service Update, multiple local SS7 boards can be configured and used under Global Call SS7 control.

1.40 Support for IP Boards and Features

The following IP boards are supported with the Service Update:

Intel NetStructure® DM/IP241-1T1-PCI-100BT
24 channels of IP gateway functionality and voice processing with single T1 span and one 100 BaseT Ethernet interface; Universal PCI form factor.

Intel NetStructure® DM/IP301-1E1-PCI-100BT
30 channels of IP gateway functionality and voice processing with single E1 span and one 100 BaseT Ethernet interface; Universal PCI form factor.

Intel NetStructure® DM/IP481-2T1-PCI-100BT
48 channels of IP gateway functionality and voice processing with dual T1 span and one 100 BaseT Ethernet interface; Universal PCI form factor.

Intel NetStructure® DM/IP601-2E1-PCI-100BT
60 channels of IP gateway functionality and voice processing with dual E1 span and one 100 BaseT Ethernet interface; Universal PCI form factor.

Intel NetStructure® DM/IP481-2T1-cPCI-100BT
48 channels of IP gateway functionality and voice processing with dual T1 span and one 100BaseT on-board data network interface; single slot; cPCI form factor.

Intel NetStructure® DM/IP601-2E1-cPCI-100BT
60 channels of IP gateway functionality and voice processing with dual E1 span and one 100BaseT on-board data network interface; single slot; cPCI form factor.

Intel NetStructure® DM/IP601-cPCI-100BT
60 channels of IP gateway functionality and voice processing with one 100 BaseT on-board data network interface; single slot; cPCI form factor.

The IP boards offer zero, one, or two T1 or E1 spans plus Voice over IP (VoIP) and media processing in a single-slot solution. These boards are ideal for larger enterprise and carrier-grade IP media gateway solutions. They support the H.323 and SIP IP telephony standards, as well as traditional APIs including R4 and Global Call. For more information about IP features supported on these boards, refer to the *Global Call IP Technology Guide*.

1.41 Support for Redundant Host (RH)

With the Service Update, Redundant Host (RH) is now supported on the following chassis/Single Board Computers (SBC):

- Intel NetStructure® ZT5085 or MPCHC5085 with MPCBL5524A1C (Dual CPU with up to 4 GB RAM)
- Intel NetStructure® ZT5085 or MPCHC5085 with MPCBL5524A1D (Single CPU with up to 4 GB RAM)

- Intel NetStructure® ZT5085 or MPCHC5085 with ZT5524A-1A (Dual CPU)
- Intel NetStructure® ZT5085 or MPCHC5085 with ZT5524A-1B (Single CPU)

Note: This release supports CompactPCI* Active/Standby Redundant Host capability. The restart mode for the components is “warm”, which means that the operating system is loaded but the Intel® software must be downloaded before the system can resume operation.

To use RH capabilities with the supported compute platforms, install the latest version of the RH software when you install the Service Update. For information about installing the RH software, refer to the *Pigeon Point Systems Linux Hot Swap Kit User Guide* located in the *redistributable-runtime/pps* directory of the release build. For information about using RH software, refer to the *Software Installation Guide*, *Administration Guide*, and *High Availability Demo Guide*.

1.42 Support for Peripheral Hot Swap (PHS) on Additional Compute Platforms

With the Service Update, Peripheral Hot Swap (PHS) is now supported on the following chassis/Single Board Computers (SBC):

- Intel NetStructure® ZT5085 or MPCHC5085 with MPCBL5524A1C (Dual CPU with up to 4 GB RAM)
- Intel NetStructure® ZT5085 or MPCHC5085 with MPCBL5524A1D (Single CPU with up to 4 GB RAM)

These are in addition to the following chassis/SBC that supported PHS prior to the Service Update:

- Intel NetStructure® ZT5085/ZT5524A-1A (Dual CPU)
- Intel NetStructure® ZT5085/ZT5524A-1B (Single CPU)
- Intel NetStructure® MPCHC5091/ZT5524A-1A
- Intel NetStructure® MPCHC5091/ZT5524A-1B
- Advantech* MIC-3081/MIC-3369
- Advantech* MIC-3038/MIC-3358
- Advantech* MIC-3041/MIC-3389

To use PHS capabilities with the supported compute platforms, install the latest version of the Hot Swap Kit (HSK) software when you install the Service Update. For information about installing the PHS software, refer to the *Pigeon Point Systems Linux Hot Swap Kit User Guide* located in the *redistributable-runtime/pps* directory of the release build. For information about using RH software, refer to the *Software Installation Guide*, *Administration Guide*, and *High Availability Demo Guide*.

1.43 New Media Load for DMV4800BC Boards

The Service Update provides a new media load, ML10C, for the Intel NetStructure® DMV4800BC Combined Media Board. This new media load provides basic voice, FSK, transaction record, and basic conferencing, with channel densities designed for use with high density background music applications.

Background music is an application where two callers chat while music plays in the background. In essence, it is a 3-party conference with the third party being a voice resource playing music. Background music applications require a ratio of 2 network interfaces to 1 voice resource to 3 conference parties. (The network interfaces are not on the DMV4800BC board; they have to come from another board, e.g., DMT160TEC.) Echo cancellation is needed in some cases but not needed in others. ML10C is for applications that do **not** require echo cancellation.

1.43.1 Feature Description

Predefined sets of features for Intel NetStructure® boards are provided in media loads. A media load consists of a configuration file set (PCD, FCD, and CONFIG files) and the associated firmware that is downloaded to the board. See the *DM3 Configuration Guide* for more information about media loads.

The features and channel densities provided by media load ML10C are as follows:

Features Supported	Basic Voice (ML1), FSK, and Transaction Record	Basic Conferencing (ML9C), No Echo Cancellation or Tone Clamping
Channel Density	120	360

There are 120 total voice resources. Any combination of the voice features (basic voice, FSK, and transaction record) can be used up to a total of 120. There are no limitations in transaction record density supporting it at full density (120). In addition to the voice resources, 360 conferencing resources can be used concurrently (without echo cancellation or tone clamping).

Conference size is limited to 60 parties without bridging. Conference bridging can be used to effectively expand a conference beyond the maximum size. Conference bridging consumes conferencing resources, reducing overall board conference density. This is not an issue for background music applications, which use 3-party conferences.

1.43.2 Configuring the Software

The new media load can be selected when using the Intel® Dialogic® Configuration Manager for Linux. The configuration procedure is described in detail in the *DM3 Configuration Guide*. This is done before the boards are started.

In addition, the following runtime parameter changes are recommended when playing music into a conference:

- When you add music to the conference, set its party attributes so that it uses transmit-only mode. That is, for the conference party that transmits music, enable the MSPA_MODEXMITONLY attribute in the MS_CDT data structure chan_attr field.
- The Conferencing AGC Noise Level Lower Threshold may have to be adjusted. This parameter is set at -40 dB by default and filters out any signals below this level. If the background music levels are low, the music may not be summed into the conference or it may come in and out. In this case, the AGC Noise Level Lower Threshold should be reduced by adding “setparm=0x3b1f, <value>” to the [0x3b] section of the CONFIG file and re-downloading the board. Further information about this parameter is given below.

CSUMS_AGC_low_threshold (AGC Noise Level Lower Threshold)

Number: 0x3B1F

Description: The **CSUMS_AGC_low_threshold** parameter defines the upper threshold for noise level estimates. Any signal above this threshold will be considered speech. Thus, this threshold should be set quite high in order to let the AGC algorithm determine when there are voiced and unvoiced periods. The parameter is given in terms of the average level.

CSUMS_AGC_low_threshold is defined as: $10^{(\text{output level in dB})/20} * 2^{23}$. Multiplying by 2^{23} converts the value into a linear 24-bit value that accommodates the 24-bit DSPs used on DM3 boards.

Values: 0x0020C5 to 0x0732AE (-60 dB to -25 dB)

Default Value: 0x0147AE (-40 dB)

Guidelines: It is recommended that the value be set in the range of -60 dB to -40 dB. Do not exceed the AGC high threshold which is set to -34.6 dB in the current DM3 system.

Here is a sample calculation to get a hexadecimal value of **CSUMS_AGC_low_threshold** for a noise threshold level of -50 dB_{avg}:

$10^{(-50/20)} * 2^{23} = 0x00679F$

1.43.3 Documentation

The online bookshelf provided with System Release 6.1 for Linux contains information about all system release features including features for application development, configuration, administration, and diagnostics.

For detailed information about configuring DMV4800BC boards, see the following document:

- *DM3 Configuration Guide*

For information about the Audio Conferencing (DCB) API, see the following documents:

- *Audio Conferencing API Programming Guide*
- *Audio Conferencing API Library Reference*

Note: The online bookshelf has not been updated for this feature, so the *DM3 Configuration Guide* does not currently include information about media load ML10C.

1.44 Dlgsnapshot Tool's Autodump Feature Now Disabled by Default

With the Service Update, the autodump feature of the dlgsnapshot tool is now disabled by default. However, you still have the option of enabling the autodump feature if you want to use it. In the default state, no board or DSP will be automatically reset as a result of a DSP failure. Previously, the entire board would be brought down if a single DSP failed. For more information about the dlgsnapshot tool, refer to the *Diagnostics Guide*.

Release Issues

The table below lists issues that can affect the hardware and software supported in Intel® Dialogic® System Release 6.1 for Linux. The following information is provided for each issue:

Issue Type

This classifies the type of release issue based on its effect on users and its disposition:

- Known – A minor hardware or software issue. This category includes interoperability issues (i.e., issues relating to combining different Intel telecom products in the same system) and compatibility issues (i.e., issues that affect the use of Intel telecom products in with third-party software or hardware). Known issues are still open but may or may not be fixed in the future.
- Known (permanent) – A known hardware or software issue or limitation that will not be fixed in the future.
- Resolved – A hardware or software issue that was resolved (usually either fixed or documented) in this release.

Defect No.

A unique identification number that is used to track each issue reported via a formal Change Control System. Additional information on defects may be available via the Defect Query tool at *http://membersresource.intel.com/defects/* (Note that when you select this link, you will be asked to either LOGIN or JOIN.)

PTR No.

Number from problem tracking system used prior to March 27, 2006. For customer convenience, both the PTR number and the corresponding defect number are shown. For issues reported after March 27, 2006, this column contains "--" and only the defect number is used to track the issue.

SU No.

For PTRs that were resolved in a Service Update, indicates the Service Update number. For PTRs that were resolved when the base release was generally available (before any Service Updates), a "--" is shown. For non-resolved issues, this information is left blank.

Product or Component

The product or component to which the issue relates, typically one of the following:

- A system-level component; for example, Host Admin
- A hardware product; for example, DM/V Boards
- A software product; for example, the Global Call library

Description

A summary description of the issue. For non-resolved issues, a workaround is included when available.

The following table lists all issues that relate to this release, sorted by Issue Type. For other sort orders, use the following links:

- View issues sorted by Service Update Number
- View issues sorted by Defect Number
- View issues sorted by Product or Component
- View issues sorted by PTR Number

Issues Sorted By Type, System Release 6.1 for Linux

Issue Type	Defect No.	PTR No.	SU No.	Product or Component	Description
Known	IPY00006147	35635		Conferencing	When running with a combination of analog and digital boards, there is a degradation in voice quality when an analog channel is added to a digital conference.
Known	IPY00006538	34708		D/300JCT-E1 Boards	When using CTR4 E1 protocol, T1 4ESS and 5ESS protocols on D/300JCT-E1 boards, the user side does not produce a “disconnected” event when the remote side does not answer. The network side does.
Known	IPY00006303	35542		DI0408LSAR2 Boards	DI0408LSAR2 with ML4 reports 100% tone/digit generation failures when running with other non-DI0408LSAR2 boards in the system and exportable resources are routed off DI0408LSAR2 to the other boards. Tone/digit generation will work on ML4 if only DI0408LSAR2 boards are in the system or if resources are not routed off the DI0408LSAR2.
Known	IPY00032251	36104		Diagnostics	its_sysinfo is referring to fwlver but unable to find it. An error message is displayed, but the failure is not reported in the sysinfo.log file.
Known	IPY00006565	34598		Diagnostics	The ISDNtrace tool consumes a portion of the CPU for each trunk that is logged and may generate an exception if CTRL-C is used to exit the tool. It may also generate an exception when invalid parameter or logical ID is entered in the command.
Known	IPY00006589	36394		DM/IP Boards	If a client is in other IP-subnetwork when using DM/IP boards, the board can’t deliver the RTP stream to the client. Also, the board can’t ping the IP address from the remote client in other IP-subnetwork.
Known	IPY00006560	36048		DM/IP Boards	The setup and retrieval of progress message along with any signal IE is not functional.
Known	IPY00006324	36564		DM/IP Boards	When running call progress PC Fax CED, it failed with the following message: Invalid Termination reason: Expected 18, Received 10 over G.711 coder.
Known	IPY00006359	36659		DM3 Network	While 2-board NFAS is configured, dropping call from makecall side in initiated state is causing firmware timeout.

Issues Sorted By Type, System Release 6.1 for Linux (Continued)

Issue Type	Defect No.	PTR No.	SU No.	Product or Component	Description
Known	IPY00029916	36581		DMN160TEC Boards	When adding DMT160TEC board on Advantech 3041/3389 with Pigeon Point Systems (PPS) installed, it fails when there are no other boards in the system. ***Workaround:*** Must have at least one board installed. It will not work with no boards installed.
Known	IPY00029918	36818		DMV160LP Boards	When the digits coming in from analog network into one of the DMV160LP front-end devices are routed over to another DMV160LP front-end device and out to the analog network again, several digits failed to be detected.
Known	IPY00005947	36714		DMV160LP Boards	With the DMV160LP board, DM3StdErr shows the following error message: Brd0 CP1: qShramDataWrite: No Node.
Known				DMV4800BC Boards	Heavy simultaneous plays or records using the 8kHz 16-bit linear coder (128 kbps) above ~ 400 channels on a DMV4800BC board with ML1B might cause quality degradation on the audio.
Known				DMV4800BC Boards	Applications requiring low latency should not use MAXTIME as a termination condition when running above ~ 420 channels on DMV4800BC with ML1B. MAXDTMF is the recommended termination condition at higher densities.
Known				DMV600BTEC Boards	On the Trunk Configuration - Specify Media Load selection screen, it lists MLFN. MLFN for 60 channel fax is not supported on DMV600BTEC board. Continue to use UL1.
Known	IPY00006271	36322		Host Admin	System intermittently receives “corrupted double-linked” message on IPT10000C boards during download.
Known	IPY00032253	35829		Host Library	When running Global Call IP applications, there may be a core dump during exit. This core dump does not affect or impact the runtime operation.
Known	IPY00010766	36131		Host Library	When running GC PDK call progress tests, unexpected GCEV_BLOCKED and GCEV_UNBLOCKED events get reported.
Known	IPY00006633	36697		Host Library	A segmentation fault occurs when RTF trace is enabled for ISDN module.
Known	IPY00006823	35851		Host Runtime Libraries	The firmware crashes when CAS_Seize is similar to the wink signal.
Known	IPY00010906	36294		IPT Series Boards	The iptfwupdate tool fails to burn new firmware to IPT Series boards. The failure message states “255 board does not recognize firmware.”
Known				IPT Series Boards	When using RFC2833, G.723 coder reports higher number of missing digit/tone failures. Most of the failures are with 1 frame per packet.

Issues Sorted By Type, System Release 6.1 for Linux (Continued)

Issue Type	Defect No.	PTR No.	SU No.	Product or Component	Description
Known				IPT Series Boards	G.729 coder reports higher number of missing digit/tone failures with VAD on.
Known				IPT Series Boards	H.323 with 'Alphanumeric' DTMF transfer mode reports higher number of missing digit/tone failures.
Known				Red Hat Linux	When running Red Hat 4.0 Update 2 Pre-emption kernel, a panic is seen occasionally during application shutdown.
Known	IPY00010862	34335		Springware Fax	Multipage fax ASCII send tests fail intermittently on D/600JCT boards. ***Workaround:*** Resend the fax if a transmission fails.
Known (permanent	IPY00008787	34722		Host Runtime	Streaming to board takes longer than expected on DM/V1200A boards. While running 1K streaming to board tests with ml2_qs2_r2mf load on DM/V1200A boards, the following error message was seen: qStreamGroupClose( ): Unbind failed fd 34, bindhandle 238, token 0x2200ee
Known (permanent)	IPY00030895	36381		2.6 Kernel	When running 2.6 kernel on ZT5085 chassis with 4 GB SBC and Intel Dialogic CompactPCI boards, there is slow OS response. The boot time exceeds 30 minutes and system is sluggish upon login. ***Workaround:*** **Solution 1** - Hiding the top 96 MB of RAM from the OS prevents access problem with pages. This can be done by adding **mem=4000M** in boot loader file for the system. For example: GRUB boot loader (RHEL4: /boot/grub/grub.conf -- SLES9: /boot/grub/menu.lst): kernel /vmlinuz <running kernel> . . . quiet **mem=4000M** LILO boot loader (/etc/lilo.conf): image=/boot/vmlinuz <running kernel> . . . append=". . . **mem=4000M** **Note:** For LILO boot loader, after editing the file, /sbin/lilo must be executed. Upon reboot, the system with Intel Dialogic CompactPCI boards will respond at normal speeds. **Solution 2** - Disable 3rd level pagetable allocation from high memory feature from the 2.6 kernel (CONFIG_HIGHPTE). Refer to the Linux distribution on how to rebuild the Linux kernel.

Issues Sorted By Type, System Release 6.1 for Linux (Continued)

Issue Type	Defect No.	PTR No.	SU No.	Product or Component	Description
Known (permanent)	IPY00010432	35500		D/300JCT and D/600JCT Boards	The DSPs on the D/300JCT and D/600JCT boards do not have enough MIPs to perform FSK processing and GTD digit processing simultaneously on the same DSP. ***Workaround:*** Perform FSK and GTD digit processing on different DSPs. For example, use channels 1-8 for FSK and channels 9-16 for GTD digit processing.
Known (permanent)	IPY00009998	33825		D/480JCT and D/600JCT Boards	For D/480JCT or D/600JCT boards, the sender should not transmit the fax at a rate greater than 9600 bps. The maximum receive rate is 9600 bps. ***Workaround:*** Set the maximum send rate to match the maximum receive rate (9600 bps).
Known (permanent)	IPY00030640	23783		DM/IP Boards	There is no interoperability between DM/IP and the Siemens IP phone when using the Embedded Stack. When working in Fast Start Mode and DM/IP is the originator, DM/IP sends facility with reason start H.245 which the Siemens IP phone does not support. ***Workaround:*** Work in Slow Start Mode.
Known (permanent)	IPY00010209	36102		DM/IP Boards	Conference density limitations to 12 conference resources on Dual Span DM/IP boards. **Note:** This does not apply to single span or resource DM/IP boards which always support the full 30 conference resources. Higher conference density can be achieved depending on the number of IP devices that are open and number of voice devices that are connected at a given time. Please see the Table , “Conference Density Limitations on Dual Span DM/IP Boards”, on page 96 for further details on what is supported.
Known (permanent)	IPY00009888	34090		DM/V Boards	When MF digits are dialed out on DM/V boards, the receiving side might get an extra “c” at the beginning of the digit string.
Known (permanent)	IPY00010332	34669		DMV4800BC Boards	Simultaneous plays on high bit rate linear coders (8kHz and 11kHz with 16-bit) may experience silence on the audio as a result of audio gaps when performing over 180 channels of simultaneous play with 8kHz or 11kHz 16-bit coder. See tech note for details: *http://resource.intel.com/telecom/support/tnotes/tnbyos/2000/tn102.htm*
Known (permanent)	IPY00006136	35891		DMV4800BC Boards	Heavy plays or records using the 11kHz 16-bit linear coder (176 kbps) above ~ 240 channels on a DMV4800BC board with ML1B might cause quality degradation on the audio due to silence gaps.

Issues Sorted By Type, System Release 6.1 for Linux (Continued)

Issue Type	Defect No.	PTR No.	SU No.	Product or Component	Description
Known (permanent)	IPY00010884	36155		Host Admin	When two IPT Series boards are installed, the boards both have the same logical ID reported by listboards -l2. The logical ID information is not available for IPT Series boards unless Peripheral Hot Swap (PHS) is active. This is unique to the IPT boards only.
Known (permanent)	IPY00030647	34043		Host Library	During application start and stop, some memory is not returned back to the system. ***Explanation:*** Linux kernel utilizes slab allocation method to manage memory. The slab allocator does not discard used memory but rather caches them. Things like process descriptors, file descriptors, etc. are cached. After an application completes, memory may not be updated in system's free memory. Linux kernel does this to optimize the restart of application by using the cached file descriptors, application data structures, etc. Usually the kernel memory flush or swap daemon, (which runs in the background) over time, will manage the unused cached memory. This is a general Linux system behavior.
Known (permanent)	IPY00028577	36198		Host Runtime Libraries	When using 2.6 kernel, a segmentation fault occurs when mixing DM3 and Springware boards in same system. It is due to the system running out of memory. 512 MB of RAM is the minimum memory required but it is a function of the memory requirements of the application. More memory may be needed.
Known (permanent)	IPY00008617	32933		Software Requirements	Intel Dialogic drivers do not support Physical Address Extensions (PAE). Users using the supported Linux version need to restrict memory to 4 GB. Intel Dialogic drivers do not support more than 4 GB of RAM.
Known (permanent)	IPY00009973	34791		TDX_UNDERRUN	When the host is not able to supply data to board at a sufficient rate, an TDX_UNDERRUN (underrun error) event will occur at about every 1.5 seconds. The delivery of data is the problem. ***Workaround:*** Ignore the extra events messages.
Resolved	IPY00030665	35955	198	Call Progress	DM3StdErr reports errors while running GC call progress.
Resolved	IPY00028444	35763	189	Call Progress	The ml2_qsa_5ESS protocol does not understand call progress message when Location field is set to 1010 on outgoing call. The firmware does not properly handle the call setup when the Location field is set to 1010.
Resolved	IPY00028338	34336	171	Call Progress Analysis	The call progress analysis comes back with false cadence connects.

Issues Sorted By Type, System Release 6.1 for Linux (Continued)

Issue Type	Defect No.	PTR No.	SU No.	Product or Component	Description
Resolved	IPY00036035	--	237	Configuration	DM/V1200BTEP fails to download E1/T1 Clear Channel protocol with "waiting for 0x9 message timed out" on Linux SR 6.1 SU232.
Resolved	IPY00035860	--	237	Configuration	DM/IP241-1T1-100BT fails to download with "waiting for 0x5 message timeout" on Linux SR 6.1 SU217.
Resolved	IPY00034678	--	241	CSP Demo	CSPAuto demo fails to return TEC_STREAM event if more than one process is run per board.
Resolved	IPY00007340	26773	--	D/82JCT-U Boards	Pressing the transfer key a second time to complete the transfer takes about 4-5 seconds to return TDX_DIAL event.
Resolved	IPY00028485	36093	189	DM/IP Boards	When running T.38 tests, some faxes in the log file are passing while others are failing with zero bytes.
Resolved	IPY00010721	36075	189	DM/IP Boards	When using ADDRESSTYPE_PHONE type aliases, it fails on H.323 Gatekeeper registration.
Resolved	IPY00009534	35090	--	DM/V Boards	When calling from JCT boards to DM/V boards, SIT is not detected.
Resolved	IPY00028463	35925	189	DM/V-A Boards	The Saturation Alarm Generation feature is not supported on DM/V-A PDK loads. This feature is still present on all DM/V-B and DMN160TEC loads, as well as all DM/V-A ISDN loads.
Resolved	IPY00010540	35853	171	DM3 Fax	Unable to send 5 page TIFF and JPEG at the same time. On DM3 boards which support fax functionality, when faxing an image following a multi-page TIFF in the same fax session, it was observed that the image did not go through and the app receives a fax error event. If the image is sent after a single page TIFF or ASCII file, the entire fax session transmission goes through.
Resolved	IPY00010022	34801	--	DM3 Fax	TFX_FAXERROR while receiving JPEG fax image.
Resolved	IPY00008556	30429	--	DM3 Fax	When sending faxes over the IP via T.38 the “TO” (FC_REMOTEID) field in the fax header is empty on half the channels.
Resolved	IPY00009451	33530	204	DM3 Firmware	After running for several weeks, half or all voice channels on DM/V2400A board are blocked. They stayed in stopped state and could not go back to idle state.
Resolved	IPY00034559	--	239	DM3 IP	Assert in the RADVISION stack.
Resolved	IPY00028657	36577	198	DM3 IP	If **ipm_stop( )** is used in the IPML application, DM3 IP cannot send out the RTP packet at all, after running IPML application the second time.
Resolved	IPY00028319	35794	171	DM3 IP	An invalid value is not returned when an out of range payload type is issued. When this occurs, **ipm_StartMedia( )** or Global Call failures are seen. Make sure that the application using payload type for RFC2833 is always between 96 and 127.

Issues Sorted By Type, System Release 6.1 for Linux (Continued)

Issue Type	Defect No.	PTR No.	SU No.	Product or Component	Description
Resolved	IPY00007718	30630	--	DM3 IP	When using G.711, the application should use the same frame size for coders as the one used by the remote side (if known).
Resolved	IPY00007640	30390	--	DM3 IP	Implementation of asymmetric coders for transmit and receive should not be used. By setting the same coder for transmit and receive in the application, asymmetric coder negotiation could be avoided.
Resolved	IPY00007555	29164	--	DM3 IP	The error message (RTP Recv: in media_RTPUnpack() SSRC failed) may occasionally be seen in DM3StdErr when running DM/IP configurations.
Resolved	IPY00010242	34320	--	DM3 Network	When SetAlarmFlow is set to have the app receive alarm at Blocking and NonBlocking, app only receives alarm during Blocking.
Resolved	IPY00008084	32160	--	DM3 Network	The progress message incorrectly sent before proceeding message.
Resolved	IPY00007908	28288	--	DM3 Network	**gc_GetSigInfo( )** fails sometimes on OFFERED event.
Resolved	IPY00007844	27539	--	DM3 Network	If a call is received on Q.931 where there is no channel ID in the setup message, the call is rejected.
Resolved	IPY00007686	31991	198	DM3 Network	When configuring inter-board NFAS where the Primary and Secondary D channel are on separate boards, NFAS trunks on the board with the Secondary D channel cannot make or accept calls. However, NFAS trunks on the Primary D channel board (intra-board NFAS) are not affected and calls can successfully be placed. If the Data Link on the Primary D channel is taken down, the Standby D channel does not sucessfully take over and now NFAS trunks on the both boards cannot make or accept calls.
Resolved	IPY00007526	25549	--	DM3 Network	During initialization sequence a DMS100 switch sets each bearer channel out of service and then issues a D-channel reset.
Resolved	IPY00007370	27563	--	DM3 Network	For DMS100 protocol after restart message, inbound calls are rejected with cause code 44 channel busy.
Resolved	IPY00007288	27764	--	DM3 Network	Outbound calls fail when the ALERTING message contains a Non-Locking Shift IE.
Resolved	IPY00007234	23614	--	DM3 Network	When a trunk receives AIS, the LineAdmin DM3 utility only displays the green and red LED lights, but not the yellow.
Resolved	IPY00007216	26043	--	DM3 Network	The NI2 firmware identifies itself as 4ESS when using a utility like ISDNTRACE.

Issues Sorted By Type, System Release 6.1 for Linux (Continued)

Issue Type	Defect No.	PTR No.	SU No.	Product or Component	Description
Resolved	IPY00007463	29364	--	DM3 Tools	Running ISDNtrace uses up all available CPU cycles.
Resolved	IPY00030894	34931	--	DM3 Voice	There's a higher percentage of record failures when all 480 channels are running the 11kHz 16-bit coder on the ML1B. When 480 channels are run with a mix of various coders, the issue does not appear.
Resolved	IPY00030654	34969	--	DM3 Voice	If **dx_play( )** is issued when using UIO, the buffer size is overwhelmed.
Resolved	IPY00009959	34011	--	DM3 Voice	ATDX_BUFDIGS is not implemented for DM3 boards.
Resolved	IPY00009619	34010	--	DM3 Voice	Function ATDX_CHNAMES( ) returns NULL pointer for DM3 boards.
Resolved	IPY00008426	33366	--	DM3 Voice	Using the io_length to specify the number of bytes to record, the dx_rec function records more bytes than expected.
Resolved	IPY00006647	36598	213	DMV1200BTEC Boards	When running Simul_Inx back-to-back on DMV1200BTEC board is causing **mem_free( )** failures in the *debugangel.log*.
Resolved	IPY00028556	35819	189	DMV1200BTEP Boards	The ML5B media load on DMV1200BTEP boards supports only 6 fax channels, although it is documented as supporting 120 transaction records in the *DM3 Configuration Guide*.
Resolved	IPY00009694	33925	--	DMV1200BTEP Boards	Stopping a simple play application using kill -9 causes the DMV1200BTEP to crash.
Resolved	IPY00028474	35874	189	DMV3600BP Boards	The UL2 media load on DMV3600BP boards supports only 34 concurrent transaction records, although it is documented as supporting 12 fax channels in the *DM3 Configuration Guide*. ***Workaround:*** Use the ML5BC or UL1 media load which supports full density transaction records.
Resolved	IPY00028581	35821	171	DMV4800BC Boards	The ML2 media load on DMV4800BC boards should be named ML2C since it provides CT Streaming of CSP data.
Resolved	IPY00009223	35046	--	DMV600BTEC Boards	The Blue LED light on the faceplate of DMV600BTEC boards does not turn on when a removebrd is issued. In addition, clicking the red snap button does not turn on the Blue LED light.
Resolved	IPY00035660	--	234	Fax	Newline character in ASCII fax corrupts received tiff file.
Resolved	IPY00034105	--	241	Fax	VFX41JCT-LS channel would become unable to send/receive fax after particular fax call scenario occurs.
Resolved	IPY00031534	--	241	Fax	When sending a fax, VFX/41JCT-LS board cannot establish phase B with some particular fax machine.

Issues Sorted By Type, System Release 6.1 for Linux (Continued)

Issue Type	Defect No.	PTR No.	SU No.	Product or Component	Description
Resolved	IPY00011037	36677	239	Fax	Inbound fax call fails. This happens when a previous call on the same device is dropped and media devices are disconnected using **gc_SetUserInfo( )**.
Resolved	IPY00034036	--	239	Gatekeeper Registration	The Gatekeeper may not respond to keep-alive registration within 2 seconds, causing GCEV_TASKFAIL at application level.
Resolved	IPY00010726	36125	189	Gateway_R4 Demo	The Gateway_R4 demo is using old libdata initialization causing **gc_Start( )** to fail when using GC_H3R_LIB cclib.
Resolved	IPY00034841	--	239	Global Call	While closing the H.323 channels, some of the H.323 channels may not close properly causing the subsequent events to be directed to incorrect devices.
Resolved	IPY00030623	29291	--	Global Call Demo	After a period of time, channels seem to stop detecting and responding to bit states/transitions.
Resolved	IPY00007364	28685	--	Global Call Demo	The Global Call demo fails when using analog protocol.
Resolved	IPY00028383	35321	--	Global Call Protocols	Busy tones are detected as “no ringback” in call progress analysis when using the **dx_dial( )** method in Global Call application.
Resolved	IPY00028378	34586	--	Global Call Protocols	For inbound call, channel is blocked after the remote caller hangs up before sending DNIS, when using pdk_hk_dtmf_io.cdp.
Resolved	IPY00010746	35042	--	Global Call Protocols	When using the pdk_us_mf_io protocol, if CDP_OUT_Send_Alerting_After_Dialing = 1 and CPA is disabled, the user expects to get the GCEV_ALERTING event right after dialing. However, if the remote side answers the call too quickly, the GCEV_CONNECTED event is returned and the GCEV_ALERTING event never comes in.
Resolved	IPY00010621	34537	--	Global Call Protocols	When using the pdk_us_mf_io protocol in the Feature Group D configuration, ANI is missing the last digit when ANI is not terminated with the expected ST digit.
Resolved	IPY00010372	35035	--	Global Call Protocols	After sending CAS_HOOKFLASH, there should be some delay before sending DTMF in pdk_sw_e1_necls_io protocol, if CDP_WaitDialToneEnabled = 0 (i.e., do not wait for dialtone).
Resolved	IPY00010223	34985	--	Global Call Protocols	pdk_sw_e1_ermx_io.cdp can only accept one ringing signal (the internal ringing or the external ringing but not both). Defining CAS_RING_APPLIED (0001 -> 0xxx) solves the detection of the two ringing signals but causes problems with outgoing calls.
Resolved	IPY00010129	34274	--	Global Call Protocols	Global Call does not provide a way to disable DISCONNECT TONE SUPERVISION with pdk_na_an_io.cdp.

Issues Sorted By Type, System Release 6.1 for Linux (Continued)

Issue Type	Defect No.	PTR No.	SU No.	Product or Component	Description
Resolved	IPY00010035	35159	--	Global Call Protocols	Under certain conditions when a **gc_MakeCall( )** attempt times out, it incorrectly displays the result message as NORMAL CLEARING instead of timeout.
Resolved	IPY00010004	34685	--	Global Call Protocols	When using the pdk_us_mf_io protocol in the Feature Group D configuration, the protocol does not send a Disconnect signal when it times out waiting for DNIS and ANI. This occurs when the remote side is configured as Feature Group B and makes a call.
Resolved	IPY00009837	35049	--	Global Call Protocols	There seems to be a hard-coded 30-second timeout on a Make Call when the call is made in Alerting mode, which will terminate the call if no one picks up the phone. The expected behavior is that the call will not be dropped automatically, so the phone will ring forever if no one picks up. This occurs on T1 CAS lines.
Resolved	IPY00009409	34663	--	Global Call Protocols	When using FXS protocol and calling a busy station using supervised transfer, you get a disconnect event for both the consultation CRN and transferred CRN.
Resolved	IPY00008220	34972	--	Global Call Protocols	When using the pdk_us_mf_io protocol, digits from the previous call are returned in ANI.
Resolved	IPY00007327	30233	--	Global Call Protocols	With the pdk_mx_r2_io protocol, if the E1 cable is disconnected and reconnected, the application does not receive all the GCEV_UNBLOCKED events.
Resolved	IPY00006809	34543	--	Global Call Protocols	When CDP_IN_DNIS_ST_Needed = 0, the pdk_e1_cas_io protocol should not issue timed-out error while waiting for DNIS.
Resolved	IPY00006804	34319	--	Global Call Protocols	If a board is configured using pdk_us_ls_fxs_io.cdp file and a call is abandoned after the first ring, the application is not receiving the GCEV_DISCONNECTED event that is expected.
Resolved	IPY00006771	34329	--	Global Call Protocols	Using Belgium R2 protocol, when configured in DTMF/MF mode, in the OFFERED state the **gc_ResetLineDev( )** function does not behave properly.
Resolved	IPY00006762	34664	--	Global Call Protocols	When using E1 line side protocol and calling a busy station using supervised transfer, you get a disconnect event for both the consultation CRN and transferred CRN.
Resolved	IPY00006748	34587	--	Global Call Protocols	The PDK E1 CAS protocol cannot be downloaded on DM3 boards, and Springware board channels cannot be opened when using this protocol.
Resolved	IPY00006735	34344	--	Global Call Protocols	On DM3, when dialtone is enabled on Belgium R2 protocol, if the first DTMF/MF digit of DNIS sent is 1 then the DNIS digits received at the inbound side are not the same as sent by the outbound side.

Issues Sorted By Type, System Release 6.1 for Linux (Continued)

Issue Type	Defect No.	PTR No.	SU No.	Product or Component	Description
Resolved	IPY00032254	36452	204	Host Admin	dlstart creates additional dlgsysmonitorserver and dlgsysclockserver processes when running more than once without dlstop in between.
Resolved	IPY00032249	36682	205	Host Admin	The VFX/41JCT-LS board fails to download on with the following error message: dxxxB1: Failed open in sctsdtdxag.c: No such file or directory.
Resolved	IPY00032055	36481	204	Host Admin	Five elements in the DLGHWINF MIB are of type integer but should be either Guage32 or Unigsned32. Errors are returned by UCD-SNMP when they are queried. Some values are overflowing to negative numbers.
Resolved	IPY00031592	36725	205	Host Admin	When running config.sh script from serial terminal, a segmentation fault is received.
Resolved	IPY00029922	35353	232	Host Admin	The listboards -l2 utility does not correctly assign the physical slot of IPT Series boards. The output shows it as unknown. For details on the physical slot, the *pmac.cfg* file contains the correct values.
Resolved	IPY00029921	36670	208	Host Admin	When running CFG, it failed to save the configuration for the D/4PCIUF board.
Resolved	IPY00028606	35713	--	Host Admin	When calling **SRLGetSubDevicesOnVirtualBoard( )** to retrieve subdevices on a virtual board will sometimes result in “error = 0x64” causing application initiation process to fail due to corrupt device names.
Resolved	IPY00028519	35685	--	Host Admin	When calling **GetClockingAgents( )** to retrieve the current TDM configuration, the results are a core dump causing application to abort.
Resolved	IPY00028399	35860	189	Host Admin	When editing the snmpd.conf file manually or using dlgcsnmpconf utility, the board server does not properly start and stop. The board server remains disabled. ***Workaround:*** Use config.sh to configure the SNMP or manually tag SNMP as configured by creating empty file /usr/dialogic/cfg/.SNMPCFG file.
Resolved	IPY00011018	36402	189	Host Admin	The dlglHidentOperStatus OID shows Springware boards that downloaded in the “failed” state.

Issues Sorted By Type, System Release 6.1 for Linux (Continued)

Issue Type	Defect No.	PTR No.	SU No.	Product or Component	Description
Resolved	IPY00010718	35554	171	Host Admin	When one of the PCD files for the DM/V2400A-PCI board is downloaded, the following warning message may appear on the console: =======< BOARD HARDWARE VERIFICATION 1======= WARNING:NO component defined in PCD file /usr/dialogic/data/ml2b_pcires.pcd for the feature sigBuf boardNumber=1 This message is harmless and can safely be ignored.
Resolved	IPY00010434	35842	189	Host Admin	When performing a single board stop, the SNMP tool (dlglHidentOperStatus OID) reports board in the failed state when the board is actually in the stopped state. ***Workaround:*** Run tblist to see if the board does not show in the list. Therefore, the board is really in stopped state.
Resolved	IPY00008347	28278	--	Host Admin	When specifying ParameterFile, it causes an error when starting Intel Dialogic services.
Resolved	IPY00008031	29850	--	Host Admin	Sigbuffer, Player, Recorder, Tonegenerator missing after download of 4x2_isdn_* loads.
Resolved	IPY00007550	35213	--	Host Admin	When stopping and then restarting the Primary Master, a segmentation fault occurs in the clocking daemon in a mixed DM3 and Springware configuration.
Resolved	IPY00032054	36657	205	Host Drivers	The drvload script runs 'cat /sbin/lsmod' to determine whether the driver is loaded, but this will never return success as it is looking inside the lsmod binary.
Resolved	IPY00029999	36817	205	Host Drivers	Error messages are displayed during dlstart because the drvload script attempts to load the Springware sctmr and dvbm drivers even though they are already loaded.
Resolved	IPY00028208	36655	198	Host Drivers	When running the Hot Swap Kit (HSK), there's issues with the install script and patch file.
Resolved	IPY00007196	36519	198	Host Drivers	The custom version of LiS included contains 6 Intel header files. Since LiS is a GPL-licensed package, the inclusion of these header files has caused them to be GPL.
Resolved	IPY00011021	36408	189	Host Install	The silent uninstall prints 2 messages on the screen, meaning that it is not completely silent.
Resolved	IPY00035831	--	239	Host Library	Segmentation fault occurs in libipm_vsc.so when calling **gc_close( )** on Global Call (IP based) line device.

Issues Sorted By Type, System Release 6.1 for Linux (Continued)

Issue Type	Defect No.	PTR No.	SU No.	Product or Component	Description
Resolved	IPY00028585	36439	198	Host Library	RESTART messages change the maintenance state of channel if the channel was IN SERVICE when message arrives.
Resolved	IPY00028547	35670	208	Host Library	PDK protocol delivers DETECTED/OFFERED event to the channel even if **gc_ReleaseCall( )** was never called to clean up the previous call on this channel.
Resolved	IPY00028352	34433	198	Host Library	**gc_MakeCall( )** fails to return any event while using the pdk_ar_r2_io protocol on E1 DM3 quad span boards.
Resolved	IPY00010854	36123	189	Host Library	When running GC call progress CEPT tests, it fails with the following invalid termination reason: 0x570 (expected 0x542)
Resolved	IPY00006827	36644	208	Host Library	When the **gc_Open( )** of analog voice device is attempted and an invalid voice device name is specified, the **gc_Open( )** fails indicating that dt_open did not work.
Resolved	IPY00006712	36790	208	Host Library	No GCEV_MEDIADETECTED event is received when the first sound heard after a connect is a SIT tone.
Resolved	IPY00030667	35861	189	Host Runtime	Streaming to board takes longer than expected on DM/V1200A boards. While running 1K streaming to board tests with ml2_qs2_r2mf load on DM/V1200A boards, the following error message was seen: qStreamGroupClose( ): Unbind failed fd 34, bindhandle 238, token 0x2200ee
Resolved	IPY00010661	33512	--	Host Runtime Libraries	The libdm3dxx library has segmentation fault when the function **dx_setevtmsk( )** is called to set on a fax channel.
Resolved	IPY00010257	33668	--	Host Runtime Libraries	On shut down, the Logger object has segmentation fault.
Resolved	IPY00007551	30437	--	Host Runtime Libraries	When dlstart is issued, a segmentation fault can occur when starting cheetah layer.
Resolved	IPY00007326	28097	--	Host Runtime Libraries	By setting up different IOTT blocks within “user I/O” an TDX_ERROR event occurs after using the function play.
Resolved	IPY00033563	--	239	IP	SIP to SIP speech path broken between 180 Ringing(SDP) and 200 OK.
Resolved	IPY00010929	36497	239	IP	GCEV_SERVICERESP is NOT received by **gc_ReqService( )** function.
Resolved	IPY00010760	36647	239	IP Host	When calling an invalid IP address or a valid one that is not answering SIP calls, you have to wait 64 seconds before you can call **gc_DropCall( )** function.

Issues Sorted By Type, System Release 6.1 for Linux (Continued)

Issue Type	Defect No.	PTR No.	SU No.	Product or Component	Description
Resolved	IPY00028380	35615	171	IP Protocols	When a Global Call H.323 application fails to register with Gatekeeper, the subsequent registrations will continue failing to register properly even if the first registration was fixed.
Resolved	IPY00028460	36298	239	IPT Series boards	An extra “reserved” codec value is sent in the INVITE if coders set to “don’t care.”
Resolved	IPY00028222	36483	239	IPT Series boards	IPT assert occurs under traffic.
Resolved	IPY00031591	36727	205	IPT Series Boards	When running on a system with IPT Series board, the DROPCALL event is missed. After the missed DROPCALL event, it issues a **gc_ReleaseCall( )** with the following error message: Function not supported in this state.
Resolved	IPY00031546	36833	205	IPT Series Boards	Unexpected IPMEV_LISTEN and IPMEV_UNLISTEN (0x9006 and 0x9007) events report are reported in the system.
Resolved	IPY00010671	35923	189	IPT Series Boards	T.38 on IPT Series boards is limited to RFC2833 and G.711 pass through. An error is reported during T.38 close when used in Inband mode even though the T.38 transmission is successful.
Resolved	IPY00028386	35532	189	Non-Facility Associated Signaling (NFAS)	The maximum number of trunks that can be controlled with one D-channel is 10. In addition, the NFAS functionality is not supported across multiple DM/V-B and DMN160TEC boards. In other words, with these two boards, NFAS can only share the D-channel providing the trunks all belong to the same board.
Resolved	IPY00010172	34732	--	Protocols	Some ISDN cause codes cannot be set using **gc_DropCall( )**.
Resolved	IPY00007236	30131	--	Protocols	GCEV_DROPCALL does not return after calling **gc_DropCall( )**.
Resolved	IPY00036247	--	241	PSTN Call Control	JCT board running with the NT1 protocol receives an Alerting message with incorrect GCEV_PROCEEDING event instead of the expected GCEV_ALERTING on a channel.
Resolved	IPY00036044	--	241	PSTN Call Control	Failures seen when invoking **gc_SetChanState( )** on JCT boards.
Resolved	IPY00034738	--	232	PSTN Call Control	Call progress analysis does not properly report fax tone when parameter All INTEGER_t CDP_OUT_ConnectType has a value of "1".
Resolved	IPY00034606	--	232	PSTN Call Control	While issuing a make call during a supervised transfer to a destination that is busy, **gc_ResultMsg( )** returns with PROTOCOL ERROR.
Resolved	IPY00033698	--	232	PSTN Call Control	The primary call can not be re-transferred via **gc_SetupTransfer( )** when the transferred call is disconnected after SwapHold.

Issues Sorted By Type, System Release 6.1 for Linux (Continued)

Issue Type	Defect No.	PTR No.	SU No.	Product or Component	Description
Resolved	IPY00032803	--	241	PSTN Call Control	Unable to make outbound calls on JCT boards with DMS protocol; the 'Interface ID Present' was being changed from 0 to 1 and the 'Interface Identifier' byte was added.
Resolved	IPY00032794	--	241	PSTN Call Control	JCT board rejects incoming calls in the first unblocked channel when a call is disconnected, but not released. This symptom was seen on incoming SETUP message with Channel ID IE specifying "ANY CHANNEL" and "host controlled release" feature is ON.
Resolved			171	Redundant Host	Redundant Host (RH) functionality is not supported in this release.
Resolved	IPY00032248	36550	204	SELinux	Users running Red Hat AS 4.0 Update 2 with SELinux enabled and a targeted policy in use will encounter problems when the Intel Dialogic start script runs at boot time. The user will be prompted multiple times during the script making unattended system boot impossible. This is due to the new default security policy present in AS 4.0 Update 2. ***Workaround:*** Edit the /usr/dialogic/bin/intel_functions script, making the following change to line 229: ORIGINAL: **su** -s /bin/bash - ${user} -c “${PROG} &>/dev/null &” MODIFIED: **/sbin/runuser** -s /bin/bash - ${user} -c “${PROG} &>/dev/null &”

Issues Sorted By Type, System Release 6.1 for Linux (Continued)

Issue Type	Defect No.	PTR No.	SU No.	Product or Component	Description
Resolved	IPY00032052	36656	205	SELinux	Systems which have the DLGCdev RPM installed will experience a kernel panic once an attempt is made to load the Springware drivers. If you have not selected “Intel Dialogic Boards” from the installation menu (DLGCdev RPM not present on the system - verify by running ‘rpm -q DLGCdev’), then it will allow unattended automatic startup at boot time as expected. This problem is due to interactions between the Linux Streams (LiS) software and the SELinux security policy. ***Workaround:*** Change the security mode from enforcing (which causes the panic) to permissive. The impact is that security issues will still be logged, but the policy will not actually be enforced. The steps to make this change are as follows: 1. If the System Release is already installed and kernel panics are occurring during system boot, reboot the system into single-user mode. If the System Release is not yet installed, skip this step. 2. Execute “/usr/sbin/setenforce 0.” This will change the security mode of the currently running kernel to permissive, but will be lost upon reboot. 3. Edit the /etc/selinux/config file changing “SELINUX=enforcing” to “SELINUX=permissive.” This will change the security mode which is set at boot time. 4. If the system was booted into single-user mode, type ‘exit’ to continue with the normal boot process. To determine the SELinux security mode, execute “/usr/sbin/sestatus | grep mode.”
Resolved	IPY00035822	--	239	SIP Call Control	Global Call SIP application does not respond to 407 Proxy Authentication Required messages.
Resolved	IPY00034627	--	239	SIP Call Control	Format Specific Parameters of the Media Format Attribute are not sent within SDP body of an Invite message when using non-default value for IPPARM_DTMF_RFC2833_PAYLOAD_TYPE.
Resolved	IPY00034406	--	239	SIP Call Control	The handle count of the application is high as compared to the previous Service Updates. The handle count used by application is around 20000.
Resolved	IPY00033912	--	239	SIP Call Control	Early media on 180 Ringing(NO SDP) fails if T.38 required.
Resolved	IPY00008148	27990	--	SNMP	If SNMP is installed, ISDN calls does not receive disconnect messages.
Resolved	IPY00034495	--	241	Springware Fax	Firmware crash occurs when certain TIFF file is sent from one channel in MH, 9600 MSLT=10ms condition.
Resolved	IPY00033185	--	241	Springware Firmware	On Springware ISDN 5ESS and 4ESS protocols, loopback calls from user to network fail.

Issues Sorted By Type, System Release 6.1 for Linux (Continued)

Issue Type	Defect No.	PTR No.	SU No.	Product or Component	Description
Resolved	IPY00035506	--	241	Springware ISDN	An ISDN call disconnects during the ACCEPT state. When this occurs the application does not get a CCEV_DISCONNECT event.
Resolved	IPY00008747	31924	--	Springware Voice	A segmentation fault results when user-defined I/O (DX_UIO) is used, and a WAVE is specified as the XPB file format when calling **dx_reciottdata( )**.
Resolved	IPY00008202	27163	--	Springware Voice	When an application issues **ec_reciottdata( )** or **ec_stream( )** on a CSP capable device and then plays a prompt using **dx_playiottdata( )**, there's a glitch that can be heard at the beginning of the play.
Resolved	IPY00008014	31925	--	Springware Voice	If DX_UIO struct is defined, any call to **dx_reciottdata( )** uses the UIO even if not specified in the IOTT struct.
Resolved	IPY00008668	50087	--	SS7 Boards	If bearer trunks are disconnected when calls are being set up, the GCEV_BLOCKED event is not generated.
Resolved	IPY00028419	35888	171	Trunk Configurator	When creating a mixed protocol load via the trunk configurator for the DMT160TEC, a current restriction exists for the E1 Clear Channel protocol, E1CC. This only affects the mixed loads that contain the E1 Clear Channel protocol. The rest of the supported protocol combinations are still functional.
Resolved	IPY00010337	34184	--	UUI IE	UUI information is not retrieved correctly with more than 71 bytes of information in the IE_BLK.
Resolved	IPY00007895	29545	--	UUI IE	UUI IE information is not retrieved correctly with more than 71 bytes of information in the IE_BLK.

Conference Density Limitations on Dual Span DM/IP Boards

Connected Voice Devices \ Open IP Devices	42	43	44	45	46	47	48	49	50	51	52	53	54	55	56	57	58	59	60
42																			30
43																		30	29
44																	30	29	28
45																30	29	28	27
46															30	29	28	27	26
47														30	29	28	27	26	25
48													30	29	28	27	26	25	24
49												30	29	28	27	26	25	24	23
50											30	29	28	27	26	25	24	23	22
51										30	29	28	27	26	25	24	23	22	21
52									30	29	28	27	26	25	24	23	22	21	20
53								30	29	28	27	26	25	24	23	22	21	20	19
54							30	29	28	27	26	25	24	23	22	21	20	19	18
55						30	29	28	27	26	25	24	23	22	21	20	19	18	17
56					30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
57				30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15
58			30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14
59		30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13
60	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12

Above table defines the Conference density limitations on Dual Span DM/IP (DM/IP601-2E1-PCI-100BT & DM/IP601-2E1-100cPCI) boards depending on the number of IP devices that are open and number of voice devices that are connected (i.e., dx_Listen( ) has been called) at a given time. Limits do not apply to single span or resource DM/IP boards which always support the full 30 conference resources.

Note: Shaded areas support full conference density of 30.

Documentation Updates

This chapter contains information on updates and corrections to the documents included in Intel® Dialogic® System Release 6.1 for Linux. Documentation updates are divided into the following categories:

3.1 System Release Documentation Updates

This section contains updates to the following documents (click the title to jump to the corresponding section):

- Release Guide

3.1.1 Release Guide

This section contains corrections and updates to the Release Guide.

Updates to Chapter 2, "System Requirements"

Section 2.2, "Basic Software Requirements"

With the Service Update, Note 3 should be revised to say that this release now supports 12 GB RAM (no longer limitation of 4 GB RAM).

Updates to Chapter 3, "New Features by Product"

Section 3.2, "New Intel Dialogic DMV160LP Products"

In the list of features for DMV160LP products, the following feature (under "Global Call support") is not supported and should be deleted (PTR# 36105):

- Hook-flash through the Global Call API

Section 3.4 "New Intel NetStructure DMT160TEC Digital Telephony Interface Products"

The following feature should be added under the "Features" heading:

- Software selectable T1/E1 (in groups of 4 spans) providing the ability to mix T1 and E1 on a board. Refer to the *DM3 Configuration Guide* for more information.

The "Call Progress Analysis" reference should be included in the following feature description (highlighted in bold):

- Tone detection and generation, including GTD/GTG and **Call Progress Analysis**. Refer to the *DM3 Configuration Guide* for more information.

Section 3.5 "New Intel NetStructure DM/V-B Products"

The use of the term "selected" is incorrect in this section. The boards only have the ability to mix "select" protocols on a board. "Selected" implies any protocol selected can be mixed; however, only a specific or a "select" number of protocols can be mixed on a board. Any reference to "selected" protocols should be changed to "select" protocols.

Section 3.6 "New Intel NetStructure IPT10000C Board"

The term "two and 1-Gigabit" should be revised to "two 100/1000 Base-T" in the "IPT10000C" description.

Section 3.9 "New Features for Intel Dialogic JCT Products"

The note under the "Continuous speech processing (CSP) support" feature is incorrect and should be ignored. The note should state the following:

Note: CSP is supported on all JCT digital boards except for the D/300JCT-E1 board.

The note under the "DPNSS support" feature is incorrect and should be ignored. The note should state the following:

Note: This feature only applies to the D/300JCT-E1, D/600JCT-1E1 and D/600JCT-2E1 boards.

The note under the "NFAS (non-facility associated signaling) support" feature is incorrect and should be ignored. The note should state the following:

Note: This feature only applies to the D/240JCT-T1, D/480JCT-1T1 and D/480JCT-2T1 boards. NFAS is only supported on T1 JCT boards and does not support E1, resource, or digital boards.

Section 3.10 "New Features for Intel NetStructure DM/F Products"

This section incorrectly states that the Intel NetStructure DM/F boards support voice. However, these boards do not provide voice support.

Section 3.12 "New Features for Intel NetStructure DM/V and DM/V-A Products"

The following feature should be added under the "New DM/V Product Features" heading:

- Additional GCAMS (Global Call Alarm Management System) alarms, which are provided through Global Call. See the *Global Call ISDN Technology Guide*, *Global Call E1/T1 CAS/R2 Technology Guide* and *DM3 Configuration Guide* for more information.

GCAMS is documented as a DM/V-A feature, but it is actually supported on DM/V boards and should be listed under that section.

The use of the term "selected media loads" is incorrect in the "Ability to stream echo-cancelled data over the CT bus to another board" feature. "Selected" implies that any

media load selected can be mixed; however, only specific or “select” number of media loads can be mixed. It should read “select media loads” in the description.

The “DM/V2400A-PCI (media load 9C)” and “DM/V2400A-cPCI (media load 5)” references should be included in the following feature description (highlighted in bold):

- Enhanced voice + fax media loads added for DM/V960A-4T1-PCI (media load 5), DM/V480A-2T1-PCI (media load 5BC), DM/V600A-2E1-PCI (media load 5BC), DM/V2400A-PCI (media load 5), **DM/V2400A-PCI (media load 9C)** and **DM/V2400A-cPCI (media load 5)**. Refer to the *DM3 Configuration Guide* for more information.

Updates to Chapter 5, “OA&M Software”

Section 5.1, “Administrative Software”

With the Service Update, Peripheral Hot Swap (PHS) is now supported on the following chassis/Single Board Computers (SBC) and should be documented under the subsection named “Peripheral Host Swap (PHS)”:

- Intel NetStructure® ZT5085 or MPCHC5085 with MPCBL5524A1C (Dual CPU with up to 4 GB RAM)
- Intel NetStructure® ZT5085 or MPCHC5085 with MPCBL5524A1D (Single CPU with up to 4 GB RAM)

Section 5.1, “Administrative Software”

With the Service Update, Redundant Host (RH) is now supported on the following chassis/Single Board Computers (SBC) and should be documented under a new subsection named “Redundant Host (RH)”:

- Intel NetStructure® ZT5085 or MPCHC5085 with MPCBL5524A1C (Dual CPU with up to 4 GB RAM)
- Intel NetStructure® ZT5085 or MPCHC5085 with MPCBL5524A1D (Single CPU with up to 4 GB RAM)
- Intel NetStructure® ZT5085 or MPCHC5085 with ZT5524A-1A (Dual CPU)
- Intel NetStructure® ZT5085 or MPCHC5085 with ZT5524A-1B (Single CPU)

Updates to Chapter 8, “Supported Hardware”

Section 8.3, “Signaling Products”

With the Service Update, the following IP Boards are supported and should be documented in the list of supported boards for Signaling Products:

- Intel NetStructure® DM/IP241-1T1-PCI-100BT
- Intel NetStructure® DM/IP301-1E1-PCI-100BT
- Intel NetStructure® DM/IP481-2T1-PCI-100BT
- Intel NetStructure® DM/IP601-2E1-PCI-100BT
- Intel NetStructure® DM/IP481-2T1-cPCI-100BT
- Intel NetStructure® DM/IP601-2E1-cPCI-100BT
- Intel NetStructure® DM/IP601-cPCI-100BT

3.2 Installation and Configuration Documentation Updates

This section contains updates to the following documents (click the title to jump to the corresponding section):

- Software Installation Guide
- DM3 Configuration Guide
- IPT Series Configuration Guide
- Springware Configuration Guide
- Global Call CDP Configuration Guide

3.2.1 Software Installation Guide

The following new subsection should be added to Chapter 4, “Troubleshooting.”

Upgrading the Hot Swap Kit/Redundant Host Software after Upgrading the System Release

Whenever you upgrade a System Release, the Hot Swap Kit and Redundant Host software should be upgraded *before* you upgrade the System Release. You could experience problems using the software if you use the latest Service Update with an old version of the Hot Swap Kit/Redundant Host software. This section describes what to do if you didn't upgrade old Hot Swap Kit and Redundant Host software before upgrading the System Release.

1. Stop all services.

2. Uninstall the Hot Swap Kit. Use hsk_uninstall as described in the *Pigeon Point Systems Linux Hot Swap Kit User Guide*, which is a PDF file located in the *redistributable-runtime/PPS* directory.

3. Reboot.

Note: If ct_intel is in the boot sequence, then there will be harmless errors during the boot session (since the driver was removed during the Hot Swap Kit/Redundant Host uninstall).

5. Install the new Hot Swap Kit. Use hsk_install as described in the *Pigeon Point Systems Linux Hot Swap Kit User Guide*, which is a PDF file located in the redistributable-runtime/PPS directory. Make sure you choose the correct Single Board Computer (SBC) and chassis.

6. Reboot into the new Hot Swap Kit/Redundant Host kernel (it should be the default).

7. Install the System Release 6.1 Service Update. Remove ct_intel from executing on boot: chkconfig ct_intel off.

8. Test Switchover before making the system live (mainly check the Domain Status).

Notes:1. If Intel Dialogic Services (build 171 or older) were started on the STANDBY, then the System Monitor would be checking the board status. Since the driver does have the PCI bus, it cannot send board status messages. Therefore, the System Monitor will think it has a crashed board. The Fault Detector tries to get a SRAM dump (which come back w/ 0). This will go on every few minutes.

2. In a glare condition, if a Switchover is happening at the same time the Fault Detector is running, a panic can occur since the SRAM is not properly activated.

3. With build 181, the Fault Detector is disabled. The System Monitor will still do board status but won't inform the Fault Detector to start. This should prevent the Fault Detector from creating crashing. But there still could be a Hot Swap related crash or SBC hang due to some hardware error.

3.2.2 DM3 Configuration Guide

The following changes apply to the DM3 Configuration Guide:

Update to **Section 2.3, "Media Loads"**

Because of features introduced in the Service Update, new media loads, ML QSB-U2 and ML 10B, are available for the Intel NetStructure DMV1200BTEC Combined Media Board. These media loads should be documented in **Section 2.3**. For information about these media loads, see Section 1.35, "New Media Loads for DMV1200BTEC Boards", on page 68 of this Release Update.

Update to **Section 2.3, "Media Loads"**

Because of a feature introduced in the Service Update, a new media load, ML10C, is available for the Intel NetStructure DMV4800BC Combined Media Board. This media load should be documented in **Section 2.3**. For information about this media load, see Section 1.43, "New Media Load for DMV4800BC Boards", on page 75 of this Release Update.

Update to **Section 2.3.1.1, "Intel NetStructure DM/IP, DM/V, DM/V-A, and DM/V-B Boards"**

The first paragraph in **Section 2.3.1.1** should read:

- Intel NetStructure DM/V, DM/V-A, and DM/V-B boards are supported by media loads 1 through 10 and universal media loads 1 through 4.

Update to **Section 2.3.1.2, "DM3 Analog Voice Boards"**

The second paragraph in **Section 2.3.1.2** should read:

- Refer to Table 4 for a list of channel densities.

Update to **Section 4.5.2, "Define the Trunk Configuration"**

Step 6 in **Section 4.5.2** in the Configuration Procedures chapter has been revised as follows:

6. From the Trunk Configuration - Specify Media Load screen, select a media load by typing the number corresponding to the media load you wish to assign to this board and then press Enter.

The Trunk Configuration - Specify Protocols for the Trunks screen will then be displayed, Figure 15 or Figure 16.

Figure 15. Trunk Configuration - Specify Protocols for the Trunks Screen (DM/V-B)

```
          Trunk Configuration - Specify Protocols for the Trunks
_____________________________________________________________________________
Select trunk protocols from the following list. Refer to a protocol by its
Protocol ID. All the selected protocols must have the same Group number.
                      Number of trunks for this board: 4

Enter the trunk number(s) for the board and the protocol(s) you want to
select for each trunk in the following format:
          <Trunk Range>:<Protcol ID>,<Trunk Range>:<Protcol ID> etc.
_____________________________________________________________________________
ProtocolID ProtocolName Group | ProtocolID ProtocolName Group
_____________________________________________________________________________

1          4ESS(T1)     1     | 2          5ESS(T1)     1
3          NTT(T1)      1     | 4          NI2(T1)      1
5          DMS(T1)      1     | 6          QSIGT1(T1)   1
7          QSIGE1(E1)   1     | 8          NET5(E1)     1
9          T1CC(T1)     1     | 10         CAS(T1)      1
11         E1CC(E1)     1     | 12         R2MF(E1)     1
13         DPNSS(E1)    2     | 14         DASS2(E1)    2
_____________________________________________________________________________

(s to save, x to save & quit, q to quit) the configuration
p to return to Trunk Configuration - Specify Media Load
r to return to Modify Board Settings
? for help and ! for navigation help
Enter the Trunk Range and Protocol ID numbers: 1-2:9,3-4:10
```

Figure 16. Trunk Configuration - Specify Protocols for the Trunks Screen (DMN160TEC or DMT160TEC)

```
          Trunk Configuration - Specify Protocols for the Trunks
_____________________________________________________________________________
Select trunk protocols from the following list. Refer to a protocol by its
Protocol ID. All the selected protocols must have the same Group number.
                      Number of trunks for this board: 4
Enter the trunk number(s) for the board and the protocol(s) you want to
select for each trunk in the following format:
          <Trunk Range>:<Protcol ID>,<Trunk Range>:<Protcol ID> etc.
_____________________________________________________________________________
ProtocolID ProtocolName Group | ProtocolID ProtocolName Group
_____________________________________________________________________________
1          4ESS(T1)     1     | 2          5ESS(T1)     1
3          NTT(T1)      1     | 4          NI2(T1)      1
5          DMS(T1)      1     | 6          QSIGT1(T1)   1
7          ISDNT1CC(T1) 1     | 8          QSIGE1(E1)   1
9          NET5(E1)     1     | 10         ISDNE1CC(E1) 1
11         T1CC(T1)     1     | 12         CAS(T1)      1
13         E1CC(E1)     1     | 14         R2MF(E1)     2
```

```
(s to save, x to save & quit, q to quit) the configuration
p to return to Trunk Configuration - Specify Media Load
r to return to Modify Board Settings
? for help and ! for navigation help
Enter the Trunk Range and Protocol ID numbers: 1-2:9,3-4:10
```

The Trunk Configuration - Specify Protocols for the Trunks screen allows you to choose a protocol for each trunk on a DM/V-B board that has network interfaces, a DMN160TEC board, or a DMT160TEC board. You may assign the same protocol or different protocols to each trunk on the board, but all of the protocols must belong to the same group number.

Notes:1. The DMN160TEC board does not support CAS or R2MF.

2. When assigning different protocols to trunks on a DMT160TEC board, in which one or more trunks includes the E1CC protocol, the following considerations apply:

- A combination of ISDN, E1CC, and R2MF protocols is not supported.
- A combination of ISDN and E1CC (CC using PDK) is not supported. Instead, use ISDN and ISDNE1CC.
- A combination of R2MF and ISDNE1CC (CC using ISDN) is not supported. Instead, use R2MF and E1CC.

Update to **Section 4.6.3, "Perform Advanced TDM Bus Configuration"**

The information following completion of the board and TDM bus configuration in **Section 4.6.3** should read as follows:

```
Would you like to configure SNMP on this system (y/n, default=n)?
```

If you installed the SNMP agent software, type `y`; otherwise type `n`.

Notes: 1. Do not enter y to configure SNMP if you have not installed the SNMP agent software. If you do, the configuration procedure is aborted and you will be prompted to run the installation script (install.sh) again so you can install the SNMP agent software.

2. If you installed SNMP agent software, but enter `n`, the following messages will appear:

```
WARNING:
Configuration of the SNMP is incomplete.
Please run '/usr/dialogic//bin/dlgcsnmpconf' to configure the SNMP.
```

3. If you installed SNMP agent software and enter `y` to configure SNMP, the SNMP Agents Configuration Tool is automatically invoked when board configuration is complete.

Update to **Section 4.8, "Completing the Configuration Utility (config.sh)"**

The information in **Section 4.8** should read as follows:

When *config.sh* is complete, the following messages are displayed:

```
Configuration is now complete.

Press <Enter> to exit the configuration tool.
```

```
Before using the software, you must ensure that the Intel(R) Dialogic(R)
environment variables are set using any one of the following actions:

    (a) At a BASH shell prompt execute: source /etc/profile.d/ct_intel.sh

    (b) At a C shell prompt execute: source /etc/profile.d/ct_intel.csh

    (c) Logout and login

    (d) Reboot system

The Intel(R) Dialogic(R) system services will automatically start every
time the system is rebooted.  To start and stop system services manually,
use the dlstart and dlstop scripts found in /usr/dialogic//bin.

Notice for CompactPCI users:
============================
If you wish to use the Hot Swap Kit or Redundant Host software, and it is
not already installed on your system, please change into the PPS
directory and run the 'hsk_install.sh' script.  Once this script has
completed, a system reboot is required.
```

Note: If SNMP agent software is installed, but not configured, the following message will be displayed following the dlstart command:

```
WARNING:
Configuration of the SNMP is incomplete.
Please run 'usr/dialogic//bin/dlgcsnmpconf' to configure SNMP.
```

Update to **Section 6.8, "[0x3b] Parameters"**

Because of a feature introduced in the Service Update, a new parameter can be added to the [0x3b] section of the CONFIG file to adjust the AGC Noise Level Lower Threshold. For information about this parameter, see Section 1.43, "New Media Load for DMV4800BC Boards", on page 75 of this Release Update.

3.2.3 IPT Series Configuration Guide

There are currently no updates to this document.

3.2.4 Springware Configuration Guide

Update to **Section 3.6.3, "Perform Advanced TDM Bus Configuration"**

The information following completion of the board and TDM bus configuration in **Section 3.6.3** should read as follows:

```
Would you like to configure SNMP on this system (y/n, default=n)?
```

If you installed the SNMP agent software, type `y`; otherwise type `n`.

Notes: ***1.*** Do not enter y to configure SNMP if you have not installed the SNMP agent software. If you do, the configuration procedure is aborted and you will be prompted to run the installation script (install.sh) again so you can install the SNMP agent software.

2. If you installed SNMP agent software, but enter n, the following messages will appear:

```
WARNING:
Configuration of the SNMP is incomplete.
Please run '/usr/dialogic//bin/dlgcsnmpconf' to configure the SNMP.
```

3. If you installed SNMP agent software and enter y to configure SNMP, the SNMP Agents Configuration Tool is automatically invoked when board configuration is complete.

Update to **Section 3.8, "Completing the Configuration Utility (config.sh)"**
The information in **Section 3.8** should read as follows:

When *config.sh* is complete, the following messages are displayed:

```
Configuration is now complete.

Press <Enter> to exit the configuration tool.

Before using the software, you must ensure that the Intel(R) Dialogic(R)
environment variables are set using any one of the following actions:

    (a) At a BASH shell prompt execute: source /etc/profile.d/ct_intel.sh

    (b) At a C shell prompt execute: source /etc/profile.d/ct_intel.csh

    (c) Logout and login

    (d) Reboot system

The Intel(R) Dialogic(R) system services will automatically start every
time the system is rebooted.  To start and stop system services manually,
use the dlstart and dlstop scripts found in /usr/dialogic//bin.

Notice for CompactPCI users:
============================
If you wish to use the Hot Swap Kit or Redundant Host software, and it is
not already installed on your system, please change into the PPS
directory and run the 'hsk_install.sh' script.  Once this script has
completed, a system reboot is required.
```

Note: If SNMP agent software is installed, but not configured, the following message will be displayed following the dlstart command:

```
WARNING:
Configuration of the SNMP is incomplete.
Please run 'usr/dialogic//bin/dlgcsnmpconf' to configure SNMP.
```

Update to **"Dialogic.Cfg Parameter Reference"** chapter
In the chapter, the following changes apply:

FirmwareFile

Usage: Board parameter, optional, applies to all baseboards.

Description: Specifies the name of a firmware load file for the system software to download to the board. This firmware file takes the place of the file that is normally downloaded.

For specifying the firmware load file of the second span on boards that have two spans, use the **FirmwareFile2** parameter.

Guidelines: When you execute Genload, the file that you specify here is located according to the following sequence:

- If a full pathname is specified (for example, **FirmwareFile = /usr/dialogic/data/spandti.fwl**), that file is used.
- If only a file name is specified (for example, **FirmwareFile = spandti.fwl**) and the file is in the directory from which Genload is executed, that file is used.
- Otherwise, the default firmware file location is */usr/dialogic/data*.

The default firmware file is the file specified using the **ISDNProtocol** parameter. If the **ISDNProtocol** parameter is set to NONE, the *spanplus.fwl* file is downloaded.

For Springware boards that support Continuous Speech Processing (CSP), a special firmware file is required. To enable CSP capability for Springware boards, you must explicitly specify the CSP firmware file. See the **FirmwareFile2** parameter for a list of standard (default) and CSP-specific firmware files.

For Springware boards that support fax, a special firmware file is required. To enable DSP-fax capability for Springware boards, you must explicitly specify the fax firmware file. See the **FirmwareFile2** parameter for a list of standard (default) and fax-specific firmware files.

Note: DSP-based fax and CSP cannot be used together on the same board.

Values: The firmware load files are installed in */usr/dialogic/data* and most have the extension *.fwl*.

Default value: Without this parameter, Genload automatically selects the correct firmware file to download.

FirmwareFile2

Usage: Board parameter, optional, applies to boards with two spans (for example, D/480JCT-2T1) and to enable Continuous Speech Processing (CSP) capability or DSP-based fax on Springware boards that support these features.

Note: DSP-based fax and CSP cannot be used together on the same board.

Description: Specifies the name of a firmware load file for the system software to download to the second span of an applicable board. This firmware file takes the place of the file that is normally downloaded.

Specify the firmware load file for the first span using the **FirmwareFile** parameter.

Guidelines: For Springware boards that support CSP or DSP-based fax, a special firmware file is required. To enable CSP or fax capability for Springware boards, you must explicitly specify the CSP or fax firmware file.

For **D/480JCT-1T1** and **D/600JCT-1E1** boards, you can provide for ISDN support on one span and CSP or fax support on the other by using two separate firmware files, one for each span.

- On the first span, you can specify an ISDN protocol and then the specific firmware file required for that ISDN protocol will be automatically downloaded to the board for that span. CSP capability is not available on this span.
- On the second span, you can enable CSP or fax capability, without ISDN support, by specifying the CSP or fax firmware file for that span and setting the ISDN protocol parameter value to **none**.

  NOTE: For E-1 and T-1 boards that support CSP or fax, specifying both an ISDN protocol (with **ISDNProtocol** or **ISDNProtocol2** parameter) and a CSP or fax firmware file (with **FirmwareFile** or **FirmwareFile2** parameter) for the same span results in a download failure to that span. The Intel Dialogic System Service will not start.

Table 3 summarizes CSP and ISDN interoperability for D/480JCT-1T1 and D/600JCT-1E1 boards. Table 4 summarizes fax and ISDN interoperability for D/480JCT-1T1 and D/600JCT-1E1 boards.

Table 3. CSP and ISDN Interoperability for D/480JCT-1T1 and D/600JCT-1E1 Boards

D/480JCT-1T1 or D/600JCT-1E1	ISDN Protocol Setting	Firmware File Setting	Result
First span	None	Standard firmware file	First span does not support ISDN.
	Specific ISDN protocol selected using the **ISDNProtocol** parameter	Firmware file specific to **ISDNProtocol** parameter automatically downloaded	First span supports ISDN.
Second span	None	CSP firmware file	Second span supports CSP.

Table 4. Fax and ISDN Interoperability for D/480JCT-1T1 and D/600JCT-1E1 Boards

D/480JCT-1T1 or D/600JCT-1E1	ISDN Protocol Setting	Firmware File Setting	Result
First span	None	Standard firmware file	First span does not support ISDN.
	Specific ISDN protocol selected using the **ISDNProtocol** parameter	Firmware file specific to **ISDNProtocol** parameter automatically downloaded	First span supports ISDN.
Second span	None	Fax firmware file	Second span supports fax.

For **D/480JCT-2T1** boards, you can provide for CSP support on one span and ISDN support on the other as follows:

- For CSP on the first span and ISDN on the second span:

```
[Genload - PCI ID xx]
FirmwareFile=spcsp.fwl /*for CSP on first span*/
ISDNProtocol2=DMS /*or other ISDN protocol for ISDN on second span*/
```

- For ISDN on the first span and CSP on the second span; note that the **ISDNProtocol2** parameter must explicitly be set to **none** in this case:

```
[Genload - PCI ID xx]
#FirmwareFile= (no FirmwareFile specified for first span)
ISDNProtocol=DMS /*or other ISDN protocol for ISDN on first span*/
FirmwareFile2=spcsp.fwl /*for CSP on second span*/
ISDNProtocol2=NONE /*must be set to none or else Genload will try to force
                   ISDNProtocol value to ISDNProtocol2 and this is not
                   supported with CSP*/
```

Values: Table 5 lists both the standard (default) firmware files and the CSP firmware files for Springware boards that support the CSP feature. Table 6 lists both the standard (default) firmware files and the DSP-based fax firmware files for Springware boards that support the fax feature.

Table 5. Firmware Files for Default and CSP Configurations

	Standard (Default) Configuration		CSP Configuration	
Board Type	**Firmware File**	**Firmware File2**	**Firmware File**	**Firmware File2**
D/120JCT-LS	*spanplus.fwl*	not applicable	*spcsp.fwl*	not applicable
D/240JCT-T1	*spanplus.fwl*	not applicable	*spcsp.fwl*	not applicable
D/480JCT-2T1	*spanplus.fwl* or **ISDNProtocol** parameter value	*spanplus.fwl* or **ISDNProtocol2** parameter value	*spcsp.fwl*	*spcsp.fwl*
D/480JCT-1T1	*spanplus.fwl* or **ISDNProtocol** parameter value	*spanplus.fwl*	*spanplus.fwl* or **ISDNProtocol** parameter value	*spcsp.fwl*
D/600JCT-1E1	*spanplus.fwl* or **ISDNProtocol** parameter value	*spanplus.fwl*	*spanplus.fwl* or **ISDNProtocol** parameter value	*spe1csp.fwl*

Table 6. Firmware Files for Default and Fax Configurations

	Standard (Default) Configuration		CSP Configuration	
Board Type	**Firmware File**	**Firmware File2**	**Firmware File**	**Firmware File2**
D/120JCT-LS	*spanplus.fwl*	not applicable	*spcsp.fwl*	not applicable
D/240JCT-T1	*spanplus.fwl*	not applicable	*spcsp.fwl*	not applicable
D/480JCT-2T1	*spanplus.fwl* or **ISDNProtocol** parameter value	*spanplus.fwl* or **ISDNProtocol2** parameter value	*spcsp.fwl*	*spfax.fwl*
D/480JCT-1T1	*spanplus.fwl* or **ISDNProtocol** parameter value	*spanplus.fwl*	*spanplus.fwl* or **ISDNProtocol** parameter value	*spfax.fwl*
D/600JCT-1E1	*spanplus.fwl* or **ISDNProtocol** parameter value	*spanplus.fwl*	*spanplus.fwl* or **ISDNProtocol** parameter value	*spe1fax.fwl*

Default value: See Table 5 or Table 6.

ISDNProtocol

Usage: Global or board parameter, optional, applies to boards with a digital network interface.

Description: Specifies that the board's digital network interface should be configured for ISDN using the selected ISDN protocol.

For specifying the ISDN protocol of the second span on boards that have two spans, use the **ISDNProtocol2** parameter.

Guidelines: The ISDN PRI Protocols package (DLGCpri) is installed with the Springware Software.

If you use the **ISDNProtocol** parameter to download an ISDN protocol firmware file to a board, the **FirmwareFile** parameter must use its default value.

Note: For E-1 and T-1 boards that support Continuous Speech Processing (CSP), specifying an ISDN protocol and a CSP firmware file for the same span results in a download failure to that span. The Intel Dialogic System Service will not start.

Note: For E-1 and T-1 boards that support DSP-based fax, specifying an ISDN protocol and a fax firmware file for the same span results in a download failure to that span. The Intel Dialogic System Service will not start.

For additional information about CSP and DSP-based fax interaction with ISDN operation, see the FirmwareFile2 parameter in this section.

Values: Valid values for **ISDNProtocol** are:

NONE	No ISDN protocol is used
4ESS	AT&T 4ESS custom switch TR41449/TR41459
5ESS	AT&T 5ESS custom switch 505-900-322
CTR4	EURO-ISDN ETSI300-102
DASS2	British National BTNR-190-1985
DMS	Northern Telecom custom switch A211-1 and A211-4
DPNSS (separately ordered)	British Private Branch Exchange DASS2 extension
ETN	EURO-ISDN ETSI300-102 for T-1
ETU	EURO-ISDN ETSI300-102 for T-1
NE1	EURO-ISDN ETSI300-102
NI2	National ISDN-2 Bellcore Special Report SR-NWT-002343
NT1	T-1 Network Emulation TR41449/TR41459
NTT	Japanese National ISDN INS-Net 1500
QNT	Q.SIG ISO 11572, ISO 11574
QTE	Q.SIG ISO 11572, ISO 11574
QTN	Q.SIG ECMA-142/143 for T-1
QTU	Q.SIG ECMA-142/143 for T-1
VNNT	French National ISDN VN3 (Network Termination)

Default value: NONE (no ISDN protocol is used)

ISDNProtocol2

Usage: Board parameter, optional, applies to digital network interface boards with two spans (for example, D/480JCT-2T1).

Description: Specifies that the board's second digital network interface should be configured for ISDN using the selected ISDN protocol.

Specify the ISDN protocol for the first span using the **ISDNProtocol** parameter (which may be a global and/or board parameter).

Guidelines: The ISDN PRI Protocols package (DLGCpri) is installed with the Springware Software.

If you use the **ISDNProtocol2** parameter to download an ISDN protocol firmware file, the **FirmwareFile2** parameter must use its default value.

Note: For E-1 and T-1 boards that support Continuous Speech Processing (CSP), specifying an ISDN protocol and a CSP firmware file for the same span results in a download failure to that span. The Intel Dialogic System Service will not start.

Note: For E-1 and T-1 boards that support DSP-based fax, specifying an ISDN protocol and a fax firmware file for the same span results in a download failure to that span. The Intel Dialogic System Service will not start.

For additional information about CSP and DSP-based fax interaction with ISDN operation, see the FirmwareFile2 parameter in this section.

Values: See the ISDNProtocol parameter in this section.

Default value: NONE (no ISDN protocol is used)

3.2.5 Global Call CDP Configuration Guide

There are currently no updates to this document.

3.3 OA&M Documentation Updates

This section contains updates to the following documents (click the title to jump to the corresponding section):

- Administration Guide
- SNMP Agent Software Administration Guide
- Diagnostics Guide
- Board Management API Library Reference
- OA&M API Programming Guide
- OA&M API Library Reference

3.3.1 Administration Guide

Tcl/Tk no longer needs to be installed. Also, no symbolic links are required to simulate Tcl/Tk 8.3. The latest QScript delivers libraries (libqscript & libqscriptgx) that are used instead of Tcl/Tk.

Therefore the following changes should be made:

In Section 4.2, "QScript Utilities," remove the following text from the first paragraph:

- and is implemented using the Tcl/Tk generic scripting language. The QScript utilities require the Tcl/Tk GUI environment.

In Section 4.2, "QScript Utilities," remove Section 4.2.1.3, "Requirements for tcl."

Update to **Section 4.7, "IPT Series Firmware Update (iptfwupdate)"**

The procedure for using the IPT Series Firmware Update utility has been revised. The new procedure is included below (PTR 36343):

The following procedure provides instructions for using the *IPT Series Firmware Update* utility:

1. Prior to installing the update, check and record the version number of the current firmware file on the system by using the Getver diagnostic utility from the */data* directory:

   ```
   Getver pmac_stl.bin
   ```

   An output similar to the following is displayed:

   ```
   Version: 1.10  Build: NBD_1.10.10_B4
   ```

   Note: The value returned is the version number of the firmware file and not the version number on the board.

   Note: You may want to copy the current firmware file to a safe location as a backup.

   For more information on the Getver utility, see the *Intel® Dialogic® System Software Diagnostics Guide*.

2. Obtain the new firmware file for IPT Series boards which is located in the */data* directory under INTEL_DIALOGIC_DIR following standard installation. This firmware file is typically installed as part of a system release or a service update.

3. After installing the update, check the version number of the firmware file on the system by using the Getver diagnostic utility from the */data* directory:

   ```
   Getver pmac_stl.bin
   ```

   If the firmware file obtained in step 3 is newer than the one recorded in step 1, continue to step 4. If the firmware file obtained in step 3 is the same as the one recorded in step 1, stop here; you do not need to perform the firmware update.

4. Optional. To determine the board number if it is not already known, invoke the *IPT Series Administration* utility. The board number is used as input to -b parameter in step 7.

   ```
   pmacadmin -l
   ```

5. Check the current version number of the firmware file on the board as follows:

   5a. Open *pmac.cfg* located in the */cfg* directory.

5b. Look for the version number, a hexadecimal value, following the IPTBoardSoftwareVersion parameter. For example:

```
IPTBoardSoftwareVersion = 0x010a0a04
```

This hexadecimal value corresponds to the decimal value returned by Getver in step 1 (1.10.10.4).

6. Reset all boards in the system and stop system services by issuing dlstop.

7. Invoke the *IPT Series Firmware Update* utility to download the firmware to the board. For example:

```
iptfwupdate -f $INTEL_DIALOGIC_DIR/data/pmac_stl.bin -b 0 -t 1
```

8. Restart all boards in the system as well as system services by issuing dlstart.

9. Check the version number of the new firmware file on the board as follows:

9a. Open *pmac.cfg* located in the */cfg* directory.

9b. Look for the version number, a hexadecimal value, following the IPTBoardSoftwareVersion parameter. This hexadecimal value should correspond to the decimal value returned by Getver in step 3.

Update to **Section 4.9, "List Board Information Utility (listboards)"**

The following note at the beginning of Section 4.9 should be deleted, because there is no longer a dependency on installing the DLGCdmdev RPM in order to use listboards in a Springware-only system (PTR# 36306):

Note: You must install the DLGCdmdev RPM to use the List Board Information utility regardless of the type of board in your system. To install the DLGCdmdev RPM, select an Intel NetStructure menu item during installation of the software release (see the Software Installation Guide).

Update to **Section 4.16, "System Logging"**

The System Logging facility has been integrated with Runtime Trace Facility (RTF). The environment DLG_TRACE_LEVEL is replaced by the RTF module OAMSYSLOG. The dlgsyslogger.log and oam.log files are no longer generated and have now been replaced by RTF log files. Refer to Chapter 23, "Runtime Trace Facility (RTF) Reference" in the *Diagnostics Guide* for more details on RTF logging.

3.3.2 SNMP Agent Software Administration Guide

A new section titled “Section 1.3, SNMP Related Startup Scripts” should be added to “Chapter 1, Administration Overview” and have the following content:

This section provides information about SNMP related startup scripts. For information about installing and configuring SNMP, refer to the Software Installation Guide and Configuration Guides provided with the System Release software.

SNMP related services are snmpd, boardserver, and dlgctrapserver. The dlstart script will start all the SNMP related services if SNMP is configured. The dlstop script will stop all the SNMP related services that were started by dlstart.

The ct_intel scripts in /etc/init.d support additional command line options for managing SNMP. These options are

- snmpstart – to start SNMP related services
- snmpstop – to stop SNMP related services
- snmprestart – to restart (i.e., stop and start) the SNMP related services.
- snmpstatus – to give the status of SNMP related services.

Note that dlgcsnmpd is supported in this release, but will be deprecated in a future release because the options are supported by the ct_intel script. The mapping is as follows:

- dlgcsnmpd start maps to ct_intel snmpstart
- dlgcsnmpd stop maps to ct_intel snmpstop
- dlgcsnmpd status maps to ct_intel snmpstatus
- dlgcsnmpd restart maps to ct_intel snmprestart

3.3.3 Diagnostics Guide

Update to **Chapter 20, “Intel Telecom Subsystem Summary Tool Reference”**

Because of an enhancement in the Service Update, the *its_sysinfo.htm* file now includes a Linux Package Info section at the beginning of the file. For further information about this feature, see Section 1.27, “New Version of its_sysinfo Tool”, on page 62 of this Release Update.

3.3.4 Board Management API Library Reference

There are currently no updates to this document.

3.3.5 OA&M API Programming Guide

There are currently no updates to this document.

3.3.6 OA&M API Library Reference

There are currently no updates to this document.

3.4 Programming Libraries Documentation Updates

This section contains updates to the following documents (click the title to jump to the corresponding section):

- Audio Conferencing API Programming Guide
- Audio Conferencing API Library Reference
- CSP API Programming Guide
- CSP API Library Reference
- Digital Network Interface Software Reference
- Fax Software Reference
- Global Call API Programming Guide
- Global Call API Library Reference
- Global Call Analog Technology User's Guide
- Global Call E1/T1 CAS/R2 Technology Guide
- Global Call IP Technology Guide
- Global Call ISDN Technology Guide
- Global Call SS7 Technology Guide
- IP Media Library API for Linux and Windows Programming Guide
- IP Media Library API Library Reference
- ISDN Software Reference
- Learn Mode and Tone Set File API Software Reference
- MSI API Programming Guide
- MSI API Library Reference
- PBX Integration Board User's Guide
- PBX Integration Software Reference
- ADEPT for PBX Integration Boards User's Guide
- SRL API Programming Guide
- SRL API Library Reference
- Voice API Programming Guide
- Voice API Library Reference

3.4.1 Audio Conferencing API Programming Guide

Update to **Chapter 6, "Application Development Guidelines"**

The following information should be added to Chapter 6, Application Development Guidelines, in a new section called "DTMF Detection Considerations" (PTR# 31134):

DTMF Detection is done separately on voice and DCB conferencing libraries.

If a DTMF detection is enabled for a conferee (via dcb_setdigitmsk), and the same receive timeslot is routed to a voice device in listen mode, both devices will detect the tone. Due to the difference in how both libraries detect the tone, it's possible to see the DTMF detected at different times. The DCB library will most likely detect the tone first since it uses the rising edge of the tone for detection.

The application should be aware of the timing difference and take the appropriate actions to avoid timing issues with DTMF detection between both libraries.

For instance, it was observed that for DTMF digits that are long enough, a voice device would still detect a DTMF even if routing of the voice in listen mode was done *after* the DCB already detected the tone (DCBEV_DIGIT) as substantial DTMF audio was remnant by the time the voice device started listening.

3.4.2 Audio Conferencing API Library Reference

There are currently no updates to this document.

3.4.3 CSP API Programming Guide

There are currently no updates to this document.

3.4.4 CSP API Library Reference

There are currently no updates to this document.

3.4.5 Digital Network Interface Software Reference

In Chapter 5, “Function Reference”, beginning on page 31, the following **dt_getstatistics( )** function reference page has been omitted.

dt_getstatistics( )

Name: int dt_getstatistics(a_hSrlDevice, a_statisticsList, a_mode)

Inputs:

int a_hSrlDevice	• logical board device handle (for example, dtiB1)
TSdtStatisticsList* a_statisticsList	• pointer to statistics
unsigned short a_mode	• synchronous/asynchronous

Returns: 0 for success
-1 for failure

Includes: srllib.h
dtilib.h

Category: Statistics Functions

Mode: synchronous/asynchronous

Description

The **dt_getstatistics( )** function returns the statistics queried. The application must specify the type of statistics to be queried in the m_StatisticsType field in the TsdtStatisticsList structure. The m_nStatisticsCount field specifies the number of statistics returned. The statistics are available as an array of TSdtLayer1Statistics structures. In asynchronous mode (EV_ASYNC) the list of statistics is part of the event data.

Parameter	Description
a_hSrlDevice	SRL handle for logical board device
a_statisticsList	pointer to TSdtStatisticList structure
a_mode	EV_SYNC or EV_ASYNC

The **dt_getstatistics( )** function uses the following data structures either directly or indirectly:

- dtStatisticsType, which is defined as follows:

```
typedef enum
{
    dtStatisticsType_Invalid = 0, /* No statistics to be collected */
    dtStatisticsType_Layer1,      /* All Layer 1 Statistics */
    dtStatisticsType_Max
}dtStatisticsType;
```

- dtStatisticsMode, which is defined as follows:

```
typedef enum
{
    dtStatisticsMode_Invalid = 0, /* No statistics Mode */
    dtStatisticsMode_Clear,       /* Clear statistics counters */
    dtStatisticsMode_Preserve,    /* Preserve statistics counters */
    dtStatisticsMode_Max
}dtStatisticsMode;
```

- dtLayer1StatisticsId, which is defined as follows:

```
typedef enum
{
    dtLayer1StatisticsId_Invalid = 0,
    dtLayer1StatisticsId_LCV = 1,    /* Line Coding Violations(LCV)           */
    dtLayer1StatisticsId_PCV,        /* Path Coding Violations(PCV)           */
    dtLayer1StatisticsId_ES,         /* Errored Seconds(ES)                   */
    dtLayer1StatisticsId_SES,        /* Severely Errored Seconds(SES)         */
    dtLayer1StatisticsId_UAS,        /* Unavailable Seconds(UAS)              */
    dtLayer1StatisticsId_BES,        /* Bursty Errored Seconds(BES)           */
    dtLayer1StatisticsId_LOFC,       /* Loss of Frame Count(LOFC)             */
    dtLayer1StatisticsId_CSS,        /* Controlled Slip Seconds(CSS)          */
    dtLayer1StatisticsId_SEFS,       /* Severly Errored Framing Seconds(SEFS) */
    dtLayer1StatisticsId_LES,        /* Line Errored Seconds(LES)             */
    dtLayer1StatisticsId_Max
}dtLayer1StatisticsId;
```

- TSdtLayer1Statistics, which is defined as follows:

```
typedef struct SdtLayer1Statistics
{
    dtLayer1StatisticsId    m_Layer1StatisticsId;
    unsigned int            m_nIntervalTotal;
    unsigned int            m_nCurrentIntervalTimer;
    unsigned int            m_nCurrentValue;
    unsigned int            m_nPreviousValue;
}TSdtLayer1Statistics;
```

- TSdtStatisticsList, which is defined as follows:

```
typedef struct SdtStatisticsList
{
   unsigned int m_nVersion; /* Version of this structure */
   dtStatisticsType  m_StatisticsType;   /* Statistics Type  */
   dtStatisticsMode  m_StatisticsMode;   /* Statistics Mode  */
   unsigned int m_nStatisticsCount;      /* Statistics Count */
   union
   {
        TSdtLayer1Statistics  m_Layer1Statistics[dtLayer1StatisticsId_Max];
   }m_Stats;
} TSdtStatisticsList;
```

Cautions

None.

Errors

Possible errors for this function include:

EDT_INVTS
: Invalid DTI device handle

EDT_PARAMERR
: Invalid parameter

EDT_TMOERR
: Synchronous function timed out waiting for reply

Example

```
/* OS Header Files */
#ifdef WIN32
#include <windows.h>
#include <process.h>    /* _beginthread, _endthread */
#include <conio.h>
#else
#include <unistd.h>
#endif
#include <stdio.h>
#include <iostream.h>
#include <iomanip.h>
#include <errno.h>
#include <stddef.h>
#include <stdlib.h>
#include <string.h>
#include <signal.h>
#include <sys/timeb.h>
#include <time.h>

/* Dialogic Header Files */
#include <gcip.h>
#include <gclib.h>
#include <gcisdn.h>
#include <dtilib.h>
#include <srllib.h>

bool repeat = true;     /* Global repeat flag and video variable */
bool EventReceived = true;
LINEDEV a_LineDev=0;
LINEDEV a_BoardDev=0;
void sig_hdlr(int temp);
void OpenBoard(void);

long EventHandler (unsigned long temp)
{
    unsigned int Loop=0;
    int dev=sr_getevtdev();
    long event=sr_getevttype();
    TSdtStatisticsList* myStatisticsList=(TSdtStatisticsList*) sr_getevtdatap();;
    printf("DevH = %d Event = 0x%X\n",dev,event);
    if(event==DTEV_GETSTATISTICS)
    {
        printf("TSdtStatisticsList - Version(%d) StatisticsType(0x%X) Count(%d) Mode(%d)\n",
            myStatisticsList->m_nVersion,myStatisticsList->m_StatisticsType,
            myStatisticsList->m_nStatisticsCount,myStatisticsList->m_StatisticsMode);

        for(Loop=0;Loop<myStatisticsList->m_nStatisticsCount;Loop++)
        {
            printf("TSLayer1Statistics(%d) - Version(%d) StatisticsId(%d) IntervalTotal(%d)
CurrentIntervalTimer(%d) CurrentValue(%d) PreviousValue(%d)\n",
                Loop,
                myStatisticsList->m_Stats.m_Layer1Statistics[Loop].m_nVersion,
                myStatisticsList->m_Stats.m_Layer1Statistics[Loop].m_Layer1StatisticsId,
                myStatisticsList->m_Stats.m_Layer1Statistics[Loop].m_nIntervalTotal,
                myStatisticsList->m_Stats.m_Layer1Statistics[Loop].m_nCurrentIntervalTimer,
                myStatisticsList->m_Stats.m_Layer1Statistics[Loop].m_nCurrentValue,
                myStatisticsList->m_Stats.m_Layer1Statistics[Loop].m_nPreviousValue);
        }
        EventReceived=true;
    }
    return 0;
}
```

```
int main(void)
{
        /* Start GlobalCall */
    signal(SIGINT,(void (*)(int))sig_hdlr);
    signal(SIGTERM,(void (*)(int))sig_hdlr);
    if (gc_Start(NULL) != GC_SUCCESS) {
        printf("gc_Start(startp = NULL) Failed\n");
        exit(0);
    }
    OpenBoard();
    gc_Close(a_BoardDev);
    gc_Stop();
    return 0;
}

void sig_hdlr(int temp)
{
    cout << "Inside sig_hdlr -> Resetting repeat flag!!" << endl;
    repeat=false;
}

void OpenBoard(void)
{
    TSdtStatisticsList myStatisticsList;
    int t_NetworkBaordDev=0;
    char a_DeviceName[120];
    strcpy(a_DeviceName,":N_dtiB1:P_ISDN");
    /* Open the board device */
    if (gc_OpenEx(&a_BoardDev,a_DeviceName, EV_SYNC, 0) != GC_SUCCESS)
    {
        printf("gc_OpenEx() failed for :%s\n",a_DeviceName);
        exit(0);
    }
    else
        printf("gc_OpenEx() successful for %s- Device Handle = %d\n",
                a_DeviceName,a_BoardDev);
    Sleep(7000);
    if (sr_enbhdlr(EV_ANYDEV, EV_ANYEVT, &EventHandler) < 0)
            cout << "Error enabling the event handler" << endl;

//Query All Layer1 metrics and clear the counters after the query
    memset(&myStatisticsList,0,sizeof(TSdtStatisticsList));
    myStatisticsList.m_nVersion=TSdtStatisticsList_VERSION_0;
    myStatisticsList.m_StatisticsType=dtStatisticsType_Layer1;
    myStatisticsList.m_StatisticsMode=dtStatisticsMode_Preserve;

    while(repeat)
    {
        if(EventReceived)
        {
            Sleep(5000);
            gc_GetNetworkH(a_BoardDev,&t_NetworkBaordDev);
            EventReceived=false;
            printf("Trying dt_getstatistics\n");
            if(dt_getstatistics(t_NetworkBaordDev,&myStatisticsList,EV_ASYNC) != 0)
            {
                printf("dt_getstatistics failed on %s Error = %s\n",
                    ATDV_NAMEP(t_NetworkBaordDev),ATDV_ERRMSGP(t_NetworkBaordDev));
            repeat = false;
            }
        }
        else Sleep( 1000 );
    }
}
```

See Also

None.

In Appendix B - Message Blocks, the following Command Message Blocks were omitted:

DTCAS_CLEAR_ALL_TEMPLATE

This command clears all templates for a particular channel. The **devh** handle must be a valid DTI channel device handle. The reply message code, DTCAS_CLEAR_ALL_TEMPLATE_COMPLETE, is received in response to this command.

The typedef for the DTCAS_CLEAR_ALL_TEMPLATE structure is as follows:

```
typedef struct t_clear_all_template_msg

{
    unsigned char  msg_code;
    unsigned char  rfu;
    unsigned short template_id;
} DTCAS_CLEAR_ALL_TEMPLATE_MSG;
```

Parameter	Description
msg_code	identifies the message type and must be set to DTCAS_CLEAR_ALL_TEMPLATE
rfu	reserved; must be set to 0 for future compatibility
template_id	specifies the template identifier

DTCAS_GET_TEMPLATE

This command gets the template for a particular channel. The **devh** handle must be a valid DTI channel device handle. The reply message code, DTCAS_GET_TEMPLATE_COMPLETE, is received in response to this command.

The typedef for the DTCAS_GET_TEMPLATE structure is as follows:

```
typedef struct t_get_template_msg
{
    unsigned char  msg_code;
    unsigned char  rfu;
    unsigned short template_id;
} DTCAS_GET_TEMPLATE_MSG;
```

Parameter	Description
msg_code	identifies the message type and must be set to DTCAS_GET_TEMPLATE
rfu	reserved; must be set to 0 for future compatibility
template_id	specifies the template identifier

DTCAS_GET_NEXT_TEMPLATE

This command gets the template for a particular channel. The **devh** handle must be a valid DTI channel device handle. The reply message code, DTCAS_GET_NEXT_TEMPLATE_COMPLETE, is received in response to this command.

The typedef for the DTCAS_GET_NEXT_TEMPLATE structure is as follows:

```
typedef struct t_get_next_template_msg
{
    unsigned char  msg_code;
    unsigned char  rfu;
    unsigned short template_id;
} DTCAS_GET_NEXT_TEMPLATE_MSG;
```

Parameter	Description
msg_code	identifies the message type and must be set to DTCAS_GET_NEXT_TEMPLATE
rfu	reserved; must be set to 0 for future compatibility
template_id	specifies the template identifier

In Appendix B - Message Blocks, the following Reply Message Blocks have been omitted:

DTCAS_CLEAR_ALL_TEMPLATE_COMPLETE

This reply message is sent in response to a DTCAS_CLEAR_ALL_TEMPLATE command. The result code within the reply message block indicates the success or failure of the command. The buffer referenced by the **replymsgp** argument will contain a valid DTCAS_REPLY_MSG message block if **dt_castmgmt( )** completes successfully.

The typedef for the DTCAS_REPLY_MSG structure is as follows:

```
typedef struct t_create_reply_msg {
    unsigned char msg_code;
    unsigned char rfu;
    unsigned short template_id;
    unsigned short result;
} DTCAS_REPLY_MSG;
```

Parameter	Description
msg_code	identifies the message type; must be set to DTCAS_CLEAR_ALL_TEMPLATE_COMPLETE
rfu	reserved; must be set to 0 for future compatibility
template_id	specifies the template identifier
result	indicates the success or failure of the command. This field is set to 0 on success, or one of the following error values if the command fails: • DTCAS_ERR_TEMPLATE_NOT_DEFINED – The template was not found in the template table • DTCAS_ERR_TEMPLATE_TABLE_EMPTY – The template table is empty; no templates are defined

DTCAS_GET_TEMPLATE_COMPLETE

This reply message is sent in response to a DTCAS_GET_TEMPLATE command. The result code within the reply message block indicates the success or failure of the command. The buffer referenced by the **replymsgp** argument will contain a valid DTCAS_REPLY_MSG message block if **dt_castmgmt( )** completes successfully.

The typedef for the DTCAS_REPLY_MSG structure is as follows:

```
typedef struct t_create_reply_msg {
    unsigned char msg_code;
    unsigned char rfu;
    unsigned short template_id;
    unsigned short result;
} DTCAS_REPLY_MSG;
```

Parameter	Description
msg_code	identifies the message type; must be set to DTCAS_GET_TEMPLATE_COMPLETE
rfu	reserved; must be set to 0 for future compatibility
template_id	specifies the template identifier
result	indicates the success or failure of the command. This field set to 0 on success, or one of the following error values if the command fails: • DTCAS_ERR_TEMPLATE_NOT_DEFINED – The template was not found in the template table • DTCAS_ERR_TEMPLATE_TABLE_EMPTY – The template table is empty; no templates are defined

DTCAS_GET_NEXT_TEMPLATE_COMPLETE

This reply message is sent in response to a DTCAS_GET_NEXT_TEMPLATE command. The result code within the reply message block indicates the success or failure of the command. The buffer referenced by the **replymsgp** argument will contain a valid DTCAS_REPLY_MSG message block if **dt_castmgmt( )** completes successfully. The typedef for the DTCAS_REPLY_MSG structure is as follows:

```
typedef struct t_create_reply_msg {
    unsigned char msg_code;
    unsigned char rfu;
    unsigned short template_id;
    unsigned short result;
} DTCAS_REPLY_MSG;
```

Parameter	Description
msg_code	identifies the message type; must be set to DTCAS_GET_NEXT_TEMPLATE_COMPLETE
rfu	reserved; must be set to 0 for future compatibility

template_id	specifies the template identifier
result	indicates the success or failure of the command. This field set to 0 on success, or one of the following error values if the command fails: • DTCAS_ERR_TEMPLATE_NOT_DEFINED – The template was not found in the template table • DTCAS_ERR_TEMPLATE_TABLE_EMPTY – The template table is empty; no templates are defined • DTCAS_ERR_END_TMPL_TABLE- The next template was not found; no other templates are defined

3.4.6 Fax Software Reference

There are currently no updates to this document.

3.4.7 Global Call API Programming Guide

There are currently no updates to this document.

3.4.8 Global Call API Library Reference

Update to **gc_MakeCall( )**

On the **gc_MakeCall( )** function reference page, on page 229, there is a change to the following paragraph (PTR# 35965):

- In the asynchronous mode, if the function is successfully initiated but connection is not achieved (no GCEV_CONNECTED event returned), then the application must issue **gc_DropCall**( ) and **gc_ReleaseCallEx**( ) functions to terminate the call completely. If GCEV_TASKFAIL is received, you must issue the **gc_DropCall**( ) and **gc_ReleaseCallEx**( ) functions before you can initiate your next call.

The last sentence of this paragraph should be deleted. More complete information about error handling in asynchronous mode, including what to do when GCEV_TASKFAIL is received, appears later on page 229.

Functions not supported in this release

The following functions, which are documented in the Global Call API Library Reference, are not supported in System Release 6.1 for Linux:

- **gc_AcceptModifyCall( )**
- **gc_RejectModifyCall( )**
- **gc_ReqModifyCall( )**

In addition to their function reference pages, these functions are also mentioned in Section 1.13, Call Modification Functions, and in Table 1, Global Call Function Support by Technology.

3.4.9 Global Call Analog Technology User's Guide

There are currently no updates to this document.

3.4.10 Global Call E1/T1 CAS/R2 Technology Guide

Update to **Section 8.2.18, "gc_SetChanState( ) Variances for E1/T1 CAS/R2"**

The following note should be added to this section (PTR# 36726):

Note: When a channel is set to out-of-service state, not all protocols send the blocking pattern by default. For such protocols, a parameter in the .cdp file has to be set to the appropriate value so that the blocking pattern is sent when the channel is put out-of-service. Refer to the *Global Call Country Dependent Parameters (CDP) for PDK Protocols Configuration Guide* for more information.

3.4.11 Global Call IP Technology Guide

Update to **Section 4.3.2, "Setting Coder Information"**

The description of IP_CAPABILITY.capability data structure field (page 109) and Table 2, "Coders Supported for Intel NetStructure IPT Boards", (pages 110-111) both list eight value defines that are used to specify the GSM AMR-NB coder. The GSM AMR-NB coder is not supported in System Release 6.1 for Linux, and these values should be ignored.

Update **Chapter 9, "IP-Specific Data Structures"**

In the reference section for the IP_CAPABILITY data structure (page 389), the description of the capability field lists eight value defines that are used to specify the GSM AMR-NB coder for IPT boards. The GSM AMR-NB coder is not supported in System Release 6.1 for Linux, and these values should be ignored.

3.4.12 Global Call ISDN Technology Guide

There are currently no updates to this document.

3.4.13 Global Call SS7 Technology Guide

Section 2.1.1, "SS7 Interface Boards" incorrectly states in the note at the end of the section that "Global Call SS7 does **not** support multiple Intel NetStructure SS7 boards in the same system." Previous versions of GCSS7 supported operation with a single SS7 board in the host system. However, starting in this release, multiple local SS7 boards can be configured and used under Global Call SS7 (GCSS7) control.

GCSS7 now supports routing of SS7 signalling to and from any configured SS7 board as well as control of the connections between timeslots on the line interfaces of those boards and the telephony bus.

The following change applies to the device naming convention in order to support multiple boards: Trunk names are now assigned based on the LIU_CONFIG commands found in the config.txt file. The **board_id** parameter of the LIU_CONFIG command is used as a major index and the **liu_id** parameter is used as a minor index for sequentially assigning trunk device names to configured LIUs. The first configured LIU (typically **board_id**=0 and **liu_id**=0) is assigned the "dkB1" trunk name and subsequent configured LIUs get the next trunk names in sequence.

This revised naming convention has an impact on systems with one SPCI2S board: Previous versions of GCSS7 would have assigned the “dkB2” and “dkB3” devices names to the trunks of an SPCI2S board. Starting with this release, as the trunk names are assigned sequentially based on the LIU_CONFIG commands, the trunks of an SPCI2S board are assigned the “dkB1” and “dkB2” device names (assuming the board is the first and only board in the system).

3.4.14 IP Media Library API for Linux and Windows Programming Guide

Dynamically Changing the Transmit Time Slot on IP Media Devices

Because of a new feature in the Service Update, a new chapter about dynamically changing the transmit time slot of IP Media devices should be added. For information about this feature, see Section 1.3, “Dynamically Changing the Transmit Time Slot on IP Media Devices”, on page 14 of this Release Update.

3.4.15 IP Media Library API Library Reference

Update to IPM_PARM_INFO data structure

Because of a new feature in the Service Update, a new parameter called PARMCH_TX_TIMESLOT should be added in IPM_PARM_INFO data structure. For information about this feature, see Section 1.3, “Dynamically Changing the Transmit Time Slot on IP Media Devices”, on page 14 of this Release Update.

3.4.16 ISDN Software Reference

In the **cc_GetParmEx( )** function reference page, the last entry in Table 22, “cc_GetParmEx( ) Parameter ID Definitions” (page 145) incorrectly lists SUBADDR_NUMBER as a parameter definition. There is no such parameter definition; the correct parameter definition is SUBADDRESS_NUMBER (PTR# 35969).

In the **cc_SetParmEx( )** function reference page, beginning on page 250, the parm_id description in the parameters table (page 251) incorrectly identifies SUBADDR_NUMBER as a supported parameter. There is no such parameter; the correct parameter name is SUBADDRESS_NUMBER.

In the **cc_GetEvtMsk( )** function reference pages, Table 20, Bitmask Values, incorrectly indicates the default values for CCMSK_SERVICE_ACK and CCMSK_SETUP_ACK as “Not enabled”. The correct default values are “Enabled.”

In the **cc_GetEvtMsk( )** function reference pages, Table 20, Bitmask Values, incorrectly lists CCMSK_TERMINATE as a supported bitmask type. The CCMSK_TERMINATE bitmask type is not supported (PTR# 29203).

In the **cc_SetEvtMsk( )** function reference pages, Table 24, Bitmask Values, incorrectly indicates the default values for CCMSK_SERVICE_ACK and CCMSK_SETUP_ACK as “Not enabled”. The correct default values are “Enabled.”

The descriptions of the CCMSK_TMREXPEVENT bitmask in the **cc_GetEvtMsk( )** and **cc_SetEvtMsk( )** functions mention that the CCEV_TIMER event is generated when a Layer 3 timer expires, but there is no description of how to retrieve the Timer ID and Call ID values associated with the CCEV_TIMER event (PTR# 29036). The following text describes how to retrieve these values with the assumption that the CCEV_TIMER event has been enabled:

In the application, define a TIMER_DATA structure as follows:

```
typedef struct _TIMER_DATA {
    unsigned char tbd_1; // 0
    unsigned long CallId; // 1 2 3 4
    unsigned short TimerId; // 5 6
    unsigned short tbd_2; // 7 8
}TIMER_DATA, *PTIMER_DATA;
```

Then, retrieve the values as follows:

```
(evtdatap = sr_getevtdatap(...)
    case CCEV_TIMER:
        {
            PTIMER_DATA pData = (PTIMER_DATA)evtdatap;
            m_TimerCallId = pData->CallId;
            m_TimerId = pData->TimerId;
            Log(MSG_EVENT,"Timer: Call_id = %d, Timer expired ID = (%d) 0x%x",
                m_TimerCallId, m_TimerId);
        }
        break;
        .
        .
        .
```

The following caution should be included in the **cc_MakeCall( )** and **cc_SetCallingNumber( )** function reference pages (PTR# 28720):

- When using **cc_MakeCall( )** to make an outbound call, if the origination_phone_number field in the MAKECALL_BLK structure is set to NULL or '\0' (null string), the destination_number_plan and the destination_number_type fields in the MAKECALL_BLK structure are ignored. This precludes the option of using the **cc_SetCallingNumber( )** function to set the origination phone number and specifying a value of NULL or '\0' for the origination_phone_number field in the MAKECALL_BLK structure, when the destination number plan and the destination number type values (as specified in the destination_number_plan and destination_number_type fields in the MAKECALL_BLK structure) must be included in the outgoing message.

In the reference information for the **cc_GetDLinkState( )** function, the description paragraph should read: "The **cc_GetDLinkState( )** function retrieves the logical data link state (operable or inoperable) of the specified board device for PRI or station device for BRI.

In the description of the **state_buf** parameter for the **cc_GetDLinkState( )** function, only two possible data link states are defined: DATA_LINK_UP and DATA_LINK_DOWN. DATA_LINK_DISABLED is not a valid value (PTR# 25745).

In the **cc_MakeCall( )** function reference information, the description of the **numberstr** parameter should read: "The destination (called party's) telephone number string. The

maximum number of digits is dictated by the protocol switch specification. Users need to find out the specification limits for the protocol they wish to use, otherwise the protocol stack will reject the request to make a call. (PTR# 22842)

3.4.17 Learn Mode and Tone Set File API Software Reference

There are currently no updates to this document.

3.4.18 MSI API Programming Guide

When using HDSI boards, application developers MUST follow the recommendation outlined in Section 2.6 “Performance Considerations” of the *SRL API Programming Guide*.

3.4.19 MSI API Library Reference

Add a note to MS_CDT chan_sel field indicating that MSPN_STATION is supported on Springware products only and MSPN_TS is supported on DI, HDSI, and Springware products. (PTR# 35565)

3.4.20 PBX Integration Board User’s Guide

Update to **Section 4.2.1, Siemens ROLM Programming Requirements** (IPY00006024 = PTR# 29612)

An additional note should be added to the NOTES at the end of this section:

With all switches supporting ROLM phone 400, the asynchronous event TD42_ASYNCCHSTATUS for reporting carrier gain is only received once, when the board starts and the port is connected to the switch. If the port is disconnected and connected again, the application does not receive any other carrier loss and gain events.

3.4.21 PBX Integration Software Reference

There are currently no updates to this document.

3.4.22 ADEPT for PBX Integration Boards User’s Guide

Update to terminology in the User’s Guide

After the bullet list on page 8 in Section 2.1, all subsequent references to "*libd42mt.dll*" and "*libd42mt.lib*" should be replaced with "*D42 library*"

The D42 library is used as a collective term for Windows and Linux:

- For Windows – *libd42mt.dll* is the runtime library file (needed to run an application) and *libd42mt.lib* is the compile time library file (needed to compile the application).

- For Linux – runtime and compile time libraries are the same and the file name is *libd42.so*.

Update to **Section 1.2 "Files Changed/Added"**

The following additions should be made:

The files that were changed or added to the Intel Dialogic System Release for ADEPT Linux implementation are listed below:

File name	Added/Changed	Location
Hicom.adt	Added	/usr/dialogic/cfg
Lucent.adt	Added	/usr/dialogic/cfg
Mitel.adt	Added	/usr/dialogic/cfg
Nec.adt	Added	/usr/dialogic/cfg
Ntbcm.adt	Added	/usr/dialogic/cfg
Ntm1.adt	Added	/usr/dialogic/cfg
Rolm.adt	Added	/usr/dialogic/cfg
libd42.so	Changed	/usr/dialogic/lib
/usr/lib		
d42lib.h	Changed	/usr/dialogic/inc
d82u.fwl	Changed	/usr/dialogic/data
d82ucsp.fwl	Changed	/usr/dialogic/data
d42ucsp.fwl	Changed	/usr/dialogic/data
Lucent_2_wire.fwl	Changed	/usr/dialogic/data
Lucent_4_wire.fwl	Changed	/usr/dialogic/data
Mitel_DNIC_M430.fwl	Changed	/usr/dialogic/data
Mitel_DNIC_M420.fwl	Changed	/usr/dialogic/data
NEC_DTerm_III.fwl	Changed	/usr/dialogic/data
Nortel_BCM.fwl	Changed	/usr/dialogic/data
Nortel_Meridian_1.fwl	Changed	/usr/dialogic/data
Nortel_Norstar.fwl	Changed	/usr/dialogic/data
Siemens_Hicom.fwl	Changed	/usr/dialogic/data
Siemens_Rolm.fwl	Changed	/usr/dialogic/data
Siemens_rolm_9006.fwl	Changed	/usr/dialogic/data

Update to **Section 2.1. "The ADEPT module in D42 library"**

Replace the first two paragraphs and bullet list with the following:

The ADEPT functionality is implemented in the D42 library (*libd42mt.dll* and *libd42mt.lib* files for Windows and *libd42.so* for Linux) with the appropriate constant declared in the *d42lib.h* file.

A rebuild of the application with the new *libd42mt.lib*/*libd42.so* and *d42lib.h* is necessary to enable the ADEPT functionality.

- *libd42mt.dll* resides in:
  windows\system32 folder and <installation folder>\dialogic\lib folder.
- *libd42mt.lib* resides in:
  <installation folder>\dialogic\inc.
- *libd42.so* resides in:
  /usr/dialogic/lib and /usr/lib.
- *d42lib.h* resides in:
  <installation folder>\dialogic\inc (for Windows) and /usr/dialogic/inc (for Linux).

See Section 1.2, “New ANI/DNIS-Enabled Parsing Tool (ADEPT) for PBX Integration Boards”, on page 14 for more information.

3.4.23 SRL API Programming Guide

There are currently no updates to this document.

3.4.24 SRL API Library Reference

There are currently no updates to this document.

3.4.25 Voice API Programming Guide

The following note should be added to Section 13.1.10, “Guidelines for Creating User-Defined Tones” (PTR# 35667):

> ***Note:*** In the case where the number of tone templates is exhausted, no error is returned on dx_addtone( ) and subsequent tone detection may fail.

The following guideline should be added to Section 13.1.10, “Guidelines for Creating User-Defined Tones” (PTR# 34546):

- On DM3 boards, building and adding tones of zero frequency values to a tone template can cause firmware failures.

3.4.26 Voice API Library Reference

There are currently no updates to this document.

3.5 Demonstration Software Documentation Updates

This section contains updates to the following documents (click the title to jump to the corresponding section):

- CSP Demo Guide

- CT Bus Clock Fallback Demo Guide
- Global Call Demo Guide
- High Availability Demo Guide
- IP Media Server (Global Call) Demo Guide
- Voice Demo Guide

3.5.1 CSP Demo Guide

There are currently no updates to this document.

3.5.2 CT Bus Clock Fallback Demo Guide

There are currently no updates to this document.

3.5.3 Global Call Demo Guide

There are currently no updates to this document.

3.5.4 High Availability Demo Guide

There are currently no updates to this document.

3.5.5 IP Media Server (Global Call) Demo Guide

There are currently no updates to this document.

3.5.6 Voice Demo Guide

There are currently no updates to this document.

3.6 Pigeon Point Systems Linux Hot Swap Kit Documentation Updates

3.6.1 Pigeon Point Systems Linux Hot Swap Kit User Guide

The following is supplemental information for this document, which is a PDF file located in the *redistributable-runtime/PPS* directory.

Performing Switchover

Follow these steps to perform switchover:

1. Select “1 – RH Commands”
2. Select “3 – Switchover Commands”
3. Choose one of the following
   - 1 Execute Fully Cooperative Switchover” – Active host – shutdown all Software (drivers) before giving up the domain
   - 2 Execute Partially Cooperative Switchover” - Active host - shutdown some of Software (drivers) before giving up the domain
   - 3 Execute Forced Switchover” – Active host take the PCI bus without informing Software (drivers) before giving up the domain
4. The following should be seen:

```
Domain 0 belongs to host 9
Domain 1 belongs to host 9
Domain 0 is disconnected
Domain 1 is disconnected
Preparing for switchover to host 11
Domain 0 is disconnected
Domain 1 is disconnected
Switchover successful
```

   (This assumes 9 was ACTIVE and 11 was STANDBY.)

If there is a failure, check the state of the domains (refer to “Checking the State of Domains.”)

Checking the State of Domains

Follow these steps to check domain ownership:

1. Select “1 – RH Commands”
2. Select “1 – Domain Information”
3. Select “3 – Get Domain Ownership”

4. You should see the following:

```
Domain #0 is owned by host #9”
Domain #1 is owned by host #9”
```

or:

```
Domain #0 is owned by host #11”
Domain #1 is owned by host #11”
```

This tells you which side is ACTIVE and which is STANDBY.

If you see the following, it means that there is a Hot Swap Kit/Redundant Host configuration problem

```
Domain #0 is owned by host #9”
Domain #1 is owned by host #11”
```

(or vice versa)

If you see the above, the Intel Dialogic System Release software will fail on boards or a panic may occur. In this case, a cool boot is required. Switchover will make the problem worse even if the domains are corrected.