



Intel® Dialogic® System Software

Diagnostics Guide

August 2006



INFORMATION IN THIS DOCUMENT IS PROVIDED IN CONNECTION WITH INTEL® PRODUCTS. NO LICENSE, EXPRESS OR IMPLIED, BY ESTOPPEL OR OTHERWISE, TO ANY INTELLECTUAL PROPERTY RIGHTS IS GRANTED BY THIS DOCUMENT. EXCEPT AS PROVIDED IN INTEL'S TERMS AND CONDITIONS OF SALE FOR SUCH PRODUCTS, INTEL ASSUMES NO LIABILITY WHATSOEVER, AND INTEL DISCLAIMS ANY EXPRESS OR IMPLIED WARRANTY, RELATING TO SALE AND/OR USE OF INTEL PRODUCTS INCLUDING LIABILITY OR WARRANTIES RELATING TO FITNESS FOR A PARTICULAR PURPOSE, MERCHANTABILITY, OR INFRINGEMENT OF ANY PATENT, COPYRIGHT OR OTHER INTELLECTUAL PROPERTY RIGHT. Intel products are not intended for use in medical, life saving, life sustaining, critical control or safety systems, or nuclear facility applications.

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

This Intel® Dialogic® System Software Diagnostics Guide as well as the software described in it is furnished under license and may only be used or copied in accordance with the terms of the license. The information in this manual is furnished for informational use only, is subject to change without notice, and should not be construed as a commitment by Intel Corporation. Intel Corporation assumes no responsibility or liability for any errors or inaccuracies that may appear in this document or any software that may be provided in association with this document.

Except as permitted by such license, no part of this document may be reproduced, stored in a retrieval system, or transmitted in any form or by any means without the express written consent of Intel Corporation.

Copyright © 2002-2006, Intel Corporation

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

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

Publication Date: August 2006

Document Number: 05-1885-007

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>

Contents

	Revision History	10
	About This Publication	13
1	Diagnostics Overview	17
1.1	Diagnostics Management Console (DMC)	17
1.2	Common Diagnostic Tasks	17
1.3	Tool Classification by Problem Type	18
2	Using the Diagnostics Management Console (DMC)	21
2.1	Starting the DMC	21
2.2	Launching a Diagnostic Tool	21
2.3	Launching a Diagnostic Tool on a Remote Machine	22
2.4	Viewing a Log File	22
2.5	Specifying a Different Location for Log Files	22
3	Checking DM3 Architecture Boards	25
3.1	Preparations	25
3.2	Checking an Individual Board	25
3.3	Checking all the Boards in a System	26
3.4	Unsupported Boards	26
3.5	Tracing Firmware	26
3.6	Diagnosing a Control Processor or Signal Processor or Shared RAM Fault on a Board ..	27
4	Checking Springware Architecture Boards	29
4.1	Requirements	29
4.2	Testing Boards with UDD	30
4.3	Aborting Tests	33
4.4	Monitoring Test Status	33
4.5	Ending a UDD Session	34
4.6	Generating a Log File	34
4.7	Troubleshooting UDD and Checking Equipment	34
5	Diagnosing First Call Issues	35
5.1	Preparation	35
5.2	Monitoring the Signal	35
5.3	Monitoring the Status of Alarms	37
6	Diagnosing PSTN Protocol Issues	39
6.1	Preparation	39
6.2	Checking the Protocol Configuration	39
7	Debugging Software	45
7.1	Troubleshooting Runtime Issues	45
7.2	Collecting System Data to Diagnose an Application Failure or Crash	48
7.3	Setting Automatic Data Collection	49

7.4	Creating a System Configuration Archive	49
8	Application Monitor Reference	51
8.1	Description	51
8.2	Guidelines	51
8.3	Options	52
9	CallInfo Reference	53
9.1	Description	53
9.2	Guidelines	53
9.3	Options	54
10	CAS Trace Reference	55
10.1	Description	55
10.2	Guidelines	55
10.3	Options	55
11	DebugAngel Reference	59
11.1	Description	59
11.2	Guidelines	59
11.3	Command Line Options	59
11.4	Additional Configuration Options	60
12	Diagbrd Reference	61
12.1	Description	61
12.2	Guidelines	61
12.3	Options	61
13	Diagnostics Management Console (DMC) Reference	63
13.1	Description	63
13.2	Guidelines	64
13.3	DMC Main Window	64
13.3.1	Menu Bar	65
13.3.2	Diagnostic Application Panels	65
13.3.3	Log File Panel	65
13.4	DMC Configuration Dialog	66
14	DigitDetector Reference	69
14.1	Description	69
14.2	Guidelines	69
14.3	Options	69
15	Dlgsnapshot Reference	71
15.1	Description	71
15.2	Guidelines	71
15.3	Options	72
15.3.1	Enabling Autodump	72
15.3.2	Running Dlgsnapshot on Demand	73
15.3.3	Command Line Options	73
16	DM3post Reference	75
16.1	Description	75

16.2	Guidelines	75
16.3	Options	76
17	DM3Trace Reference	79
17.1	Description	79
17.2	Guidelines	79
17.3	Options	79
18	Getver Reference	81
18.1	Description	81
18.2	Options	81
18.3	Output	82
19	ISDN Trace Reference	83
19.1	Description	83
19.2	Guidelines	83
19.3	Options	83
20	Intel Telecom Subsystem Summary Tool Reference	85
20.1	Description	85
20.2	Command Line Interface	85
20.3	Graphical User Interface (Windows only)	86
20.4	Information Collected by its_sysinfo	86
20.5	System Information Data Structuring	88
21	KernelVer Reference	91
21.1	Description	91
21.2	Options	91
22	PDK Trace Reference	93
22.1	Description	93
22.2	Guidelines	93
22.3	Options	94
22.4	Sample Scenarios	94
23	Phone Reference	97
23.1	Description	97
23.2	Guidelines	97
23.3	Options	97
24	PSTN Diagnostics Tool Reference	99
24.1	Description	99
24.2	Guidelines	100
24.3	Preparing to Run the PSTN Diagnostics Tool	100
24.4	Running the PSTN Diagnostics Tool	100
24.4.1	Starting the PSTN Diagnostics Tool	100
24.4.2	The Main Window	101
24.4.3	The View Bar	102
24.4.4	Option Buttons and Command Buttons	102
24.4.5	Keyboard Navigation	107
24.5	Checking the pstndiag Log File	107

24.6	PSTN Diagnostics Menus	108
24.6.1	File menu	108
24.6.2	Log menu	108
24.6.3	View menu	109
24.6.4	Help menu	109
24.7	PSTN Diagnostics Command Line Options	109
25	QScript Reference	111
25.1	Description	111
25.2	Guidelines	111
25.3	QScript Environment Variables	112
26	RTFManager Reference	113
26.1	Description	113
26.2	Guidelines	114
26.3	Starting RTFManager	114
26.4	Main Window	114
26.4.1	File	115
26.4.2	Edit	115
26.4.3	Tools	116
26.4.4	Help	116
26.5	General Tab	116
26.6	Filtering Tab	118
26.7	Log File Filtering	120
26.8	Advanced Tab	121
27	Runtime Trace Facility (RTF) Reference	123
27.1	Description	123
27.2	Installing RTF	124
27.3	RTF Configuration File	124
27.3.1	RTFConfig Tag	125
27.3.2	Logfile Tag	127
27.3.3	Global Tag	129
27.3.4	GLabel Tag	130
27.3.5	GClient Tag	131
27.3.6	GClientLabel Tag	131
27.3.7	Module Tag	132
27.3.8	MLabel Tag	133
27.3.9	MClient Tag	134
27.3.10	MClientLabel Tag	134
27.3.11	Precedence Scheme for Module/Client Labels	135
27.4	Restrictions and Limitations	136
27.4.1	Guidelines for Editing the RTF Configuration File	136
27.5	rtftool Command	138
27.5.1	Pausing and Reloading RTF	138
27.5.2	Clearing the RTF Trace Log File Contents	138
27.5.3	Running RTF in Preservation Mode	138
27.6	Example RTF Configuration Files	139
27.6.1	Example 1: Tracing disabled - RtfConfigWin.xml	139
27.6.2	Example 2: Tracing enabled, logfile path and size specified, preservation mode OFF, one module configured - RTFConfigLinux.xml	139

27.6.3	Example 3: Tracing enabled, logfile path and size specified, several modules configured, global configuration used - RTFConfigWin.xml	140
27.7	Configuring Remote RTF Logging	142
27.7.1	Configuring the Client System	142
27.7.2	Configuring the Server System	143
27.7.3	Cautions	143
28	Status Monitor Reference	145
28.1	Status Monitor for Windows	145
28.1.1	Description	145
28.1.2	Guidelines	145
28.1.3	Options	146
28.2	Status Monitor for Linux	148
28.2.1	Description	148
28.2.2	Guidelines	148
28.2.3	Usage	148
29	UDD Reference	151
29.1	Description	151
29.2	User Interface Considerations	151
29.3	Error Messages from UDD Startup	152
29.4	UDD GUI Screens	153
29.4.1	Board Information Displayed on UDD Main Screen	153
29.4.2	UDD Main Screen Menu Options	154
29.4.3	UDD Main Screen Button Options	154
29.4.4	UDD Specify Tests Screen Buttons	154
29.5	UDD Tests	155
29.5.1	Board Sanity Check	155
29.5.2	Shared RAM Test	156
29.5.3	Local RAM Test	156
29.5.4	DSP Test	156
29.5.5	Board Timer Test	157
29.5.6	Onboard Interrupt Test	157
29.5.7	SC2000 Test	157
29.5.8	EEPROM Test	158
29.5.9	PC Interrupt Test	158
29.5.10	T1 Loopback Test	158
29.5.11	E1 Loopback Test	159
29.5.12	LS Frontend Test	159
29.5.13	NS Hardware Test	160
	Glossary	161
	Index	163

Figures

1	Example of UDD Main Screen	31
2	UDD Specify Tests Screen	32
3	UDD Test Progress Screen	33
4	PSTN Diagnostics Tool - Board Level View	36
5	PSTN Diagnostics Tool - Outbound Channel (Upper Pane) & Inbound Channel (Lower Pane)	40
6	PSTN Diagnostics Tool - Making a Call	41
7	PSTN Diagnostics Tool - Call Completed and Released	43
1	DMC Main Window	64
2	DMC Configuration Dialog	67
3	PSTN Diagnostics Tool - System Level View	101
4	PSTN Diagnostics Tool - View Bar	102
5	Toggle Command Buttons	103
6	Direct Command Buttons	103
7	Channel State Command Buttons	103
8	Outbound Call Control Command Buttons	104
9	Inbound Call Command Buttons	104
10	Call Hold Command Buttons	105
11	Call Transfer Command Buttons	106
12	SendISDN Command Button	106
13	Misc. Commands Buttons	106
14	External Application Command Button	107
15	RTFManager Filtering Tab	119
16	RTF Configuration File Tag Structure	125
17	RTF Configuration File Edited to Configure the Client System	142
18	RTF Configuration File Edited to Configure the Server System	143
19	Example of Status Monitor (for Windows) Output	147
20	Statusmon (for Linux) Call Mode	149
21	Statusmon (for Linux) Bit Mode	150

Tables

1	Log Files Archived by its_sysinfo.	87
2	Error Messages	152
3	UDD Main Screen Buttons.	154
4	Specify Tests Buttons	155

Revision History

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

Document No.	Publication Date	Description of Revisions
05-1885-007	August 2006	<p>Diagnostics Overview: Added new section introducing the Diagnostics Management Console (DMC).</p> <p>Using the Diagnostics Management Console (DMC): New chapter.</p> <p>Diagnostics Management Console (DMC) Reference: New chapter.</p>
05-1885-006	July 2006	<p>About This Publication: Added the new chapters to How to Use This Publication.</p> <p>Diagnostics Overview: Added references about new tools to Common Diagnostic Tasks and Tool Classification by Problem Type.</p> <p>Debugging Software: This chapter now includes a new version of Troubleshooting Runtime Issues that contains an additional procedure for using the RTFManager GUI and a new procedure for adding the new Application Monitor tool: Setting Automatic Data Collection.</p> <p>Application Monitor Reference: New chapter.</p> <p>RTFManager Reference: New chapter.</p> <p>Runtime Trace Facility (RTF) Reference: Added a note about maximum backups limit. In Restrictions and Limitations, added reference to Release Updates for information about errors in RTF log.</p>
05-1885-005	April 2006	<p>Getver Reference: Chapter revised to document the new version of this tool.</p> <p>Intel Telecom Subsystem Summary Tool Reference: Chapter revised to document the new version of this tool.</p> <p>PDK Trace Reference: In Description, added information about which boards support this tool.</p> <p>Phone Reference: Removed reference to the Audio Control tool.</p> <p>QScript Reference: Removed Audio Control tool and CAS Signal Editor from the list of administrative utilities.</p> <p>Runtime Trace Facility (RTF) Reference: Chapter revised to document the new version of this tool. Updated Module Tag section and removed Table 3 since the most current list of modules is in the RTF configuration file. Updated module and label names in RTF Configuration File and Example RTF Configuration Files. Added information to maxbackups. Removed mention of binary trace output since it is now all text output. Removed the following line from the RTF configuration file: <!DOCTYPE RTFConfig SYSTEM "RTFConfig.dtd" >.</p> <p>Status Monitor Reference: New chapter.</p>

Document No.	Publication Date	Description of Revisions
05-1885-004	August 2005	<p>Checking an Individual Board: Instructions for using the Diagbrd tool to perform this were replaced with instructions for using DM3post. Added a note explaining that Diagbrd is deprecated and will be removed in a future release.</p> <p>Diagnosing PSTN Protocol Issues: Added Preparation section.</p> <p>Debugging Software: Added two sections to this chapter: Collecting System Data to Diagnose an Application Failure or Crash and Creating a System Configuration Archive.</p> <p>Troubleshooting Runtime Issues: Added reference to requirements for tool.</p> <p>CallInfo Reference: In Guidelines, changed workaround for running QScript utilities.</p> <p>CAS Trace Reference: Added this chapter.</p> <p>Diagbrd Reference: Added a note explaining that Diagbrd is deprecated and will be removed in a future release.</p> <p>DigitDetector Reference: In Guidelines, added requirement for running QScript utilities.</p> <p>DM3post Reference: Added this chapter.</p> <p>Getver Reference: Added capability for getting version of Linux binaries.</p> <p>Intel Telecom Subsystem Summary Tool Reference: Added this chapter.</p> <p>Phone Reference: In Guidelines, changed the workaround for running QScript utilities.</p> <p>PSTN Diagnostics Tool Reference: In Guidelines, added requirement of installing the XFree86-libs and TK RPMs. Changed workaround for running QScript utilities.</p>
05-1885-004	August 2005	<p>QScript Reference: Added this chapter.</p> <p>Running the PSTN Diagnostics Tool: Expanded Option Buttons and Command Buttons to list and describe all buttons.</p> <p>Runtime Trace Facility (RTF) Reference: Added a note about the Mandrake security utility.</p> <p>Glossary: Added this chapter.</p>
05-1885-003	March 2005	<p>CallInfo Reference: Workaround for using QScript utilities added.</p> <p>Dlgsnapshot Reference: Added this chapter because this tool is now available for Linux. Added board support, fault types, MS to filename, examples of file names, and examples of commands.</p> <p>PDK Trace Reference: Added this chapter.</p> <p>Phone Reference: Workaround for using QScript utilities added.</p> <p>PSTN Diagnostics Tool Reference: Workaround for using QScript utilities added.</p> <p>Runtime Trace Facility (RTF) Reference: In Files Used by the RTF Tool, replaced RtfConfig.xml with RtfConfigLinux.xml and RtfConfigWin.xml and added RTFConfig.dtd and librtf.so.</p>

Document No.	Publication Date	Description of Revisions
05-1885-002	November 2004	<p>System Requirements chapter: Removed. This information is provided in the Release Guide.</p> <p>Checking DM3 Architecture Boards: Added this chapter.</p> <p>Checking Springware Architecture Boards: Added this chapter. This chapter, along with UDD Reference, replaces the standalone document <i>Dialogic Universal Hardware Diagnostics Guide</i> and contain information taken from it.</p> <p>Debugging Software: Added this chapter.</p> <p>Diagbrd Reference: Added -r (reset) option to Guidelines and Options.</p> <p>DebugAngel Reference: Added this chapter.</p> <p>DM3KDebug Reference: Removed this chapter.</p> <p>DM3StdErr Reference: Removed this chapter.</p> <p>PSTN Diagnostics Tool Reference: Added this chapter.</p> <p>Qerror: Removed this chapter.</p> <p>Runtime Trace Facility (RTF) Reference: Added this chapter.</p> <p>UDD Reference: Added this chapter. This chapter, along with Checking Springware Architecture Boards, replaces the standalone document <i>Dialogic Universal Hardware Diagnostics Guide</i> and contain information taken from it.</p>
05-1885-001	September 2002	<p>Initial version of document. Much of the information contained in this document was previously contained in the <i>DM3 Diagnostic Utilities Reference Guide</i>, document number 05-1484-005.</p>



About This Publication

The following topics provide information about this publication:

- [Purpose](#)
- [Intended Audience](#)
- [How to Use This Publication](#)
- [Related Information](#)

Purpose

This guide describes the diagnostic tools included with the system software release and explains how to use them.

Note: All tools described in this publication apply to Intel NetStructure® on DM3 architecture boards only, with the exception of the UDD tool (explained in [Chapter 29, “UDD Reference”](#)) and the Diagbrd tool (explained in [Chapter 12, “Diagbrd Reference”](#)). The UDD tool is intended for Intel® Dialogic® Springware architecture boards and the Diagbrd tool applies to any board that is supported by the software release.

Intended Audience

This information is intended for:

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

How to Use This Publication

Refer to this document after you have installed the hardware and the Intel® Dialogic® system software that includes the diagnostic tools.

The information in this guide is organized as follows:

- [Chapter 1, “Diagnostics Overview”](#) provides an overview of the diagnostic tools and the debugging or troubleshooting tasks that can be performed with them.

- [Chapter 2, “Using the Diagnostics Management Console \(DMC\)”](#) describes how to use the DMC GUI to launch the diagnostic tools and view the associated log files.

Each of the following chapters discusses diagnostic tasks. Details about the tools used to perform the tasks are provided in the tool reference chapters that follow these task chapters. This information is cross-referenced between the task and reference chapters.

- [Chapter 3, “Checking DM3 Architecture Boards”](#)
- [Chapter 4, “Checking Springware Architecture Boards”](#)
- [Chapter 5, “Diagnosing First Call Issues”](#)
- [Chapter 6, “Diagnosing PSTN Protocol Issues”](#)
- [Chapter 7, “Debugging Software”](#)

Each of the remaining chapters provides information about an individual diagnostic tool. The chapters are organized in alphabetical order:

- [Chapter 8, “Application Monitor Reference”](#) - The Application Monitor tool can be used to monitor an application and if the monitored application terminates abnormally, Application Monitor will launch the its_sysinfo tool to collect data that can help in diagnosing the problem.
- [Chapter 9, “CallInfo Reference”](#) - The CallInfo tool detects call information using the DM3 board’s Telephony Service Provider (TSP) resource.
- [Chapter 10, “CAS Trace Reference”](#) - The CAS Trace tool enables you to track the bit level transitions on a robbed bit or CAS line.
- [Chapter 11, “DebugAngel Reference”](#) - The DebugAngel tool provides low-level firmware tracing to aid in low-level debugging.
- [Chapter 12, “Diagbrd Reference”](#) - Diagbrd is a script that runs the DM3post tool, which can perform diagnostics on a stopped board at any time to detect and isolate possible hardware faults.
- [Chapter 13, “Diagnostics Management Console \(DMC\) Reference”](#) - The Diagnostics Management Console (DMC) provides a single portal from which you can launch the diagnostic tools supplied with Intel® telecom software. The DMC also enables you to locate the log files produced by these tools and view them with the appropriate viewer
- [Chapter 14, “DigitDetector Reference”](#) - The DigitDetector tool provides the ability to detect digits at the local end of a channel connection.
- [Chapter 15, “Dlgsnapshot Reference”](#) - The Dlgsnapshot tool uses Intel Dialogic system software fault monitoring components to generate a core dump file when a Control Processor (CP) or Signal Processor (SP), or Shared RAM (SRAM) fault is detected on a DM3 board.
- [Chapter 16, “DM3post Reference”](#) - The DM3post tool allows you to perform diagnostics on a stopped board at any time to detect and isolate possible hardware faults. DM3post also provides an option to run POST on all the DM3 architecture boards in a chassis.

Note: Because the DM3post tool now has more functionality than the Diagbrd tool and can be used on both Windows and Linux, the Diagbrd tool will be deprecated in a future release.
- [Chapter 17, “DM3Trace Reference”](#) - The DM3Trace tool acts as a virtual serial port (terminal emulation) session to the board.

- [Chapter 18, “Getver Reference”](#) - The getver (Get Version) software application outputs version information for files that are part of the Intel® Telecom software installation.
- [Chapter 19, “ISDN Trace Reference”](#) - The ISDNtrace tool provides the ability to track Layer 3 (Q.931) messages on the ISDN D-channel. ISDNtrace prints messages on the screen in real time.
- [Chapter 20, “Intel Telecom Subsystem Summary Tool Reference”](#) - the Intel® Telecom Subsystem Summary Tool (its_sysinfo) provides a simple way to collect information about systems built using Intel® telecom products. The its_sysinfo tool provides information about the system environment: the operating system, computer architecture, Intel Dialogic System Release, and operational logs.
- [Chapter 21, “KernelVer Reference”](#) - The KernelVer tool can be used to verify whether or not a processor has crashed.
- [Chapter 22, “PDK Trace Reference”](#) - The DM3 PDK Protocol Trace (PDK Trace) tool allows those who use a DM3 PDK protocol to log specific information related to the operation of the protocol.
- [Chapter 23, “Phone Reference”](#) - The Phone tool uses the TSC and ToneGen instances and requires a TSC component. The Phone tool can control a single DM3 resource channel (make calls, wait for calls etc.), monitor channel and call states, and send call control operations to a DM3 Global Call resource.
- [Chapter 24, “PSTN Diagnostics Tool Reference”](#) - The PSTN Diagnostics tool (pstndiag) is a utility for diagnosing and troubleshooting public switched telephone network (PSTN) connectivity problems on specific hardware products based on DM3 architecture.
- [Chapter 25, “QScript Reference”](#) - The QScript utilities (CallInfo, DigitDetector, Phone, and PSTN Diagnostics) are a subset of the diagnostic utilities. This chapter provides supplemental information about this group of tools.
- [Chapter 26, “RTFManager Reference”](#) - The Runtime Trace Facility Manager GUI (RTFManager) can be used to configure the trace levels and output of the Runtime Trace Facility (RTF) utility. If you don’t use this GUI, you must edit the RTF configuration file manually.
- [Chapter 27, “Runtime Trace Facility \(RTF\) Reference”](#) - The RTF tool provides a mechanism for tracing the execution path of various runtime libraries. The resulting log file/debug stream output helps troubleshoot runtime issues for applications that are built with Intel Dialogic software.
- [Chapter 28, “Status Monitor Reference”](#) - The Status Monitor tool enables you to track the state of the TSC as well as the state of the bits on a robbed bit or CAS line.
- [Chapter 29, “UDD Reference”](#) - UDD is a diagnostic utility used for testing the functionality of Intel Dialogic boards with Springware architecture.

Related Information

Refer to the following documents and Web sites for more information:

-
- Administration Guide for the system release
- *SNMP Agent Software for Linux Operating Systems Administration Guide*



- Configuration Guides for the system release
- Release Guide
- Release Update
- <http://developer.intel.com/design/telecom/support/> (for technical support)
- <http://www.intel.com/design/network/products/telecom/index.htm> (for product information)

This chapter presents an overview of the diagnostic tools and the debugging/troubleshooting tasks that can be performed with them. The following topics are included:

- [Diagnostics Management Console \(DMC\)](#) 17
- [Common Diagnostic Tasks](#) 17
- [Tool Classification by Problem Type](#)..... 18

1.1 Diagnostics Management Console (DMC)

The Diagnostics Management Console (DMC) provides a single portal from which you can launch all the diagnostic tools supplied with Intel® telecom software. The DMC also enables you to locate the log files produced by these tools and view them with the appropriate viewer.

The DMC allows you to launch the diagnostic tools remotely through the standard remote control methods provided with the operating system, such as SSH or Remote Desktop.

For more information about the DMC, refer to the following chapters:

- [Chapter 2, “Using the Diagnostics Management Console \(DMC\)”](#)
- [Chapter 13, “Diagnostics Management Console \(DMC\) Reference”](#)

1.2 Common Diagnostic Tasks

This section provides a list of diagnostic tasks that can be performed by using the various diagnostics tools. Another way to find the tool and procedure you need is to look at problem types. Refer to [Section 1.3, “Tool Classification by Problem Type”](#), on page 18.

Hardware and Firmware Diagnostics

The following tasks can be accomplished using the diagnostic tools:

Checking the hardware

You can use the Diagbrd, DM3post, KernelVer, and UDD tools to troubleshoot the physical boards in your system. Refer to these sections of the manual. Refer to the following chapters:

- [Chapter 3, “Checking DM3 Architecture Boards”](#)
- [Chapter 4, “Checking Springware Architecture Boards”](#)
- [Chapter 12, “Diagbrd Reference”](#)
- [Chapter 21, “KernelVer Reference”](#)
- [Chapter 29, “UDD Reference”](#)

Checking the firmware

You can use the DebugAngel tool to diagnose and trace problems with a board's firmware. Refer to the following:

- [Section 3.5, “Tracing Firmware”](#), on page 26
- [Chapter 11, “DebugAngel Reference”](#)

Software and Application Diagnostics

The following tasks can be accomplished using the diagnostic tools:

Checking the network connections

You can use the PSTN Diagnostics, Digit Detector, and CallInfo tools to check your board's network connections. Refer to the following:

- [Chapter 5, “Diagnosing First Call Issues”](#)
- [Chapter 24, “PSTN Diagnostics Tool Reference”](#)
- [Chapter 9, “CallInfo Reference”](#)

Checking the PSTN protocol configuration

You can use the PSTN Diagnostics tool to check your board's protocol configuration. Refer to the following:

- [Chapter 6, “Diagnosing PSTN Protocol Issues”](#)
- [Chapter 24, “PSTN Diagnostics Tool Reference”](#)

Collecting system data

You can use the Intel® Telecom Subsystem Summary Tool (its_sysinfo) to collect information about systems built using Intel® telecom products. Refer to the following:

- [Section 7.2, “Collecting System Data to Diagnose an Application Failure or Crash”](#), on page 48
- [Section 7.3, “Setting Automatic Data Collection”](#), on page 49
- [Section 7.4, “Creating a System Configuration Archive”](#), on page 49
- [Chapter 20, “Intel Telecom Subsystem Summary Tool Reference”](#)
- [Chapter 8, “Application Monitor Reference”](#)

Debugging software

You can use the Runtime Trace Facility (RTF) tool to trace the execution path of various runtime libraries. Refer to the following:

- [Section 7.1, “Troubleshooting Runtime Issues”](#), on page 45
- [Chapter 27, “Runtime Trace Facility \(RTF\) Reference”](#)
- [Chapter 26, “RTFManager Reference”](#)

1.3 Tool Classification by Problem Type

This section groups the diagnostic tools by problem type. The list that follows can help you locate the appropriate tool to use based on the problem you are having:

Board, Firmware, or Hardware Failures

To troubleshoot the physical boards in your system, use the following procedures and tools:

- [Chapter 3, “Checking DM3 Architecture Boards”](#)
- [Chapter 4, “Checking Springware Architecture Boards”](#)
- [Chapter 12, “Diagbrd Reference”](#)
- [Chapter 15, “Dlgsnapshot Reference”](#)
- [Chapter 21, “KernelVer Reference”](#)
- [Chapter 29, “UDD Reference”](#)

To diagnose and trace problems with a board’s firmware, use the following procedures and tools:

- [Section 3.5, “Tracing Firmware”](#), on page 26
- [Chapter 11, “DebugAngel Reference”](#)

PSTN Connectivity

You can use the PSTN Diagnostics, Digit Detector, and CallInfo tools to check your board’s network connections. Refer to the following:

- [Chapter 5, “Diagnosing First Call Issues”](#)
- [Chapter 24, “PSTN Diagnostics Tool Reference”](#)
- [Chapter 9, “CallInfo Reference”](#)

Protocol Issues

Refer to the following:

- [Chapter 6, “Diagnosing PSTN Protocol Issues”](#)
- [Chapter 24, “PSTN Diagnostics Tool Reference”](#)
- [Chapter 22, “PDK Trace Reference”](#)

Software Issues

You can use the Runtime Trace Facility (RTF) tool for some general software debugging. Refer to the following:

- [Chapter 7, “Debugging Software”](#)
- [Chapter 27, “Runtime Trace Facility \(RTF\) Reference”](#)
- [Chapter 26, “RTFManager Reference”](#)

You can use the Intel Telecom Subsystem Summary Tool (its_sysinfo) to collect information about systems built using Intel telecom products. Refer to the following:

- [Section 7.2, “Collecting System Data to Diagnose an Application Failure or Crash”](#), on page 48
- [Section 7.3, “Setting Automatic Data Collection”](#), on page 49
- [Section 7.4, “Creating a System Configuration Archive”](#), on page 49
- [Chapter 20, “Intel Telecom Subsystem Summary Tool Reference”](#)
- [Chapter 8, “Application Monitor Reference”](#)



Using the Diagnostics Management Console (DMC)

2

This chapter describes how to use the Diagnostics Management Console (DMC) to launch the diagnostic tools supplied with Intel® telecom software and view the log files produced by the tools. The following information is included:

- [Launching a Diagnostic Tool](#) 21
- [Launching a Diagnostic Tool on a Remote Machine](#) 22
- [Viewing a Log File](#) 22
- [Specifying a Different Location for Log Files](#) 22

For a description of the DMC and its windows and menus, refer to [Chapter 13, “Diagnostics Management Console \(DMC\) Reference”](#).

2.1 Starting the DMC

On Windows systems, you can start the DMC via the Start menu: **Start** > [**Intel telecom software release**] > **Diagnostics Management Console (DMC)**. The DMC application can be found in the *bin* directory of the installed Intel telecom software.

On Linux systems, you can start the DMC from anywhere since the *bin* directory is part of the path.

The DMC application (*DMC.jar*) is the same for both Windows and Linux. You use *DMC.bat* to launch the DMC on Windows and *dmc.sh* to launch the DMC on Linux.

2.2 Launching a Diagnostic Tool

Locate the tool you want in the list of local diagnostic applications (on the left side of the DMC main window), click on it to highlight it, and then press **Enter** or select **Execute application** from the **File** menu or press **F5**. You can also double-click on the tool name to launch it.

The DMC will execute the application via the command shell.

Related information:

- [Section 2.3, “Launching a Diagnostic Tool on a Remote Machine”](#), on page 22
- [Section 2.4, “Viewing a Log File”](#), on page 22
- [Section 13.3.3, “Log File Panel”](#), on page 65

2.3 Launching a Diagnostic Tool on a Remote Machine

To use a diagnostics tool on a remote machine, you must change settings on the [DMC Configuration Dialog](#).

1. Access the [DMC Configuration Dialog](#) by selecting **Configuration** from the **Tools** menu *or* pressing **F4**.
2. Check the box next to **Remote Execution of CLI Tools**.
3. Enter the **Remote Machine Name**.
4. Enter the **Remote Login UserName**.
5. In the **Remote Tool Path** field, specify the path of the diagnostic tool that you will use remotely via a CLI. You can use the **Browse Path** button to specify the path.
6. Click **OK**.
7. Locate the tool you want in the list of remote diagnostic applications (on the left side of the DMC main window), click on it to highlight it, and then press **Enter** *or* select **Execute application** from the **File** menu *or* press **F5**. You can also double-click on the tool name to launch it.

Related information:

- [Section 13.4, “DMC Configuration Dialog”](#), on page 66
- [Section 2.2, “Launching a Diagnostic Tool”](#), on page 21 (local machine)
- [Section 2.4, “Viewing a Log File”](#), on page 22
- [Section 13.3.3, “Log File Panel”](#), on page 65

2.4 Viewing a Log File

To view a log file, locate the file you want in the log file list, click on it to highlight it, and then press **Enter** *or* select **Open log file** from the **File** menu *or* use the **CTRL+O** shortcut. You can also double-click on the log file name to view it. The viewer that is associated with a given log file and located on the local machine will be used to view the log file.

For more information about locating log files and the log file list, refer to [Section 13.3.3, “Log File Panel”](#), on page 65.

2.5 Specifying a Different Location for Log Files

You can use the **Browse** button on the log file panel to locate log files. However, if you want to change the default location in which the DMC will look for log files, you can specify it on the DMC Configuration Dialog. You can access the DMC Configuration Dialog by selecting

Configuration from the **Tools** menu or pressing **F4**. A **Browse** button allows you to select a directory for the Log File Directory field.

Note: If the user-defined variable is empty or not valid, the DMC will use the environment variable set up during installation of the Intel telecom software: INTEL_DIALOGIC_DIR/log.

For more information about the DMC's list of log files, refer to [Section 13.3.3, "Log File Panel"](#), on page 65.



Checking DM3 Architecture Boards

3

This chapter provides procedures for checking the Intel NetStructure® on DM3 architecture hardware and firmware in your system.

- Preparations 25
- Checking all the Boards in a System 26
- Unsupported Boards 26
- Tracing Firmware 26
- Diagnosing a Control Processor or Signal Processor or Shared RAM Fault on a Board27

3.1 Preparations

To find out which boards are in your system and which slots they are in, on Windows systems you can use the Configuration Manager (DCM) and on Linux systems you can run `./config.sh` to get the board configuration tool. In both cases, the main menu will provide information about the boards.

3.2 Checking an Individual Board

This section provides a procedure for using the DM3post tool to check an individual DM3 architecture board at any time to detect and isolate possible hardware faults. For more information on this tool, refer to [Chapter 16, “DM3post Reference”](#).

Note: Linux users can also use the Diagbrd script for this task. It is used in the same way as DM3post. However, because the DM3post tool now has more functionality than the Diagbrd tool and can be used on both Windows and Linux, the Diagbrd tool will be deprecated in a future release. For more information about Diagbrd, refer to [Chapter 12, “Diagbrd Reference”](#).

To use the DM3post tool to check a DM3 board, follow this procedure:

1. Make sure the board is in a “Stopped” state. This is necessary because DM3post will reset the specified board, forcing the Control Processor (CP) POST diagnostics to run.

DM3post will then retrieve the POST results from the SRAM and provide a PASS/FAIL indication. The board will remain in a stopped state and you will be required to restart the board.

2. Run the DM3post command, specifying the slot and bus number of the board. For example, the following command runs the DM3post tool on a board in slot 17, bus 0:

```
dm3post -s17 -b0
```

3. You will get the following response:

```
Do you wish to continue (y/n)?
```

If you answer Y, the following will be printed to the screen:

```
dm3post processing...
```

The success/failure message will be printed to the screen when POST is complete.

3.3 Checking all the Boards in a System

The DM3post diagnostic utility provides an option to run POST on a chassis level. By using the chassis option (-c), DM3post will retrieve the results of the last run POST for all DM3 boards in the chassis. By using the chassis option (-c) with the reset (-r) option, you can run POST on all DM3 boards in the system.

When using the chassis option, it is not necessary to provide the bus and slot numbers. Any option other than the reset option will be ignored when using the chassis option.

In addition to output on the screen, more detailed output is logged to a log file, *dm3post.log*, by default.

3.4 Unsupported Boards

The Intel® Dialogic® System Release software will not prevent you from installing an unsupported board. However, the configuration manager will not show any unsupported boards. An error message about the unsupported board(s) will appear in a log file in the *log* directory with the following filename: *rtf*.txt* (for example, *rtflog-10072005-14h47m25.639s.txt*). The following is a sample error message:

```
10/04 14:55:50.784 1792 1820 OAMSYSLOG ErrorEx NCMAPI - Board bus-1,
slot-9, model-256, serialNumber-KU005523 is not supported in this release
```

3.5 Tracing Firmware

This section describes how to use the DebugAngel tool to perform low-level firmware tracing for low-level debugging. For more information about the DebugAngel tool, refer to [Chapter 11, “DebugAngel Reference”](#).

DebugAngel polls the DM3 boards in the system and posts **qPrintf()** statements from the resources and DM3 kernel to a log file.

To use DebugAngel, follow these steps:

1. Install the service and start it running in automatic mode by entering the following command:

```
DebugAngel -install
```

2. When you need to refer to the firmware trace log, look in the *log* directory:

- On a Windows system, information is logged to the file *DebugAngel.log* in the *%INTEL_DIALOGIC_DIR%\log* directory.
- On a Linux system, information is logged to the file *debugangel.log* in the *\${INTEL_DIALOGIC_DIR}/log* directory.

Command line options and configuration options for DebugAngel are provided in [Chapter 11](#), “DebugAngel Reference”.

3.6 Diagnosing a Control Processor or Signal Processor or Shared RAM Fault on a Board

This section describes how the Dlgsnapshot tool is used to capture a core dump when a Control Processor (CP) or Signal Processor (SP), or Shared RAM (SRAM) fault is detected on a DM3 board. For details about the Dlgsnapshot tool, refer to [Chapter 15](#), “Dlgsnapshot Reference”.

When a fault occurs, the board on which the fault occurred is stopped automatically without interrupting the system and a core dump file is created in the *log* directory. The naming convention for this file is described in [Chapter 15](#), “Dlgsnapshot Reference”.

Windows systems: When a core dump file is generated, you can send a *.zip* file containing the contents of the *%INTEL_DIALOGIC_DIR%\log* directory along with the Windows event viewer system log file (**.evt*) to Intel’s telecom support resources for debugging purposes.

Linux systems: When a core dump file is generated, you can send a *.tar* file containing the contents of the *\${INTEL_DIALOGIC_DIR}/log* directory to Intel’s telecom support resources for debugging purposes.

You can also run the Dlgsnapshot on demand from the command line. You must specify the print buffer to be dumped. Other command line options include the AUID, processor number, logical ID, physical slot number, PCI bus number, and PCI slot number. A description of all the command line options is given in [Chapter 15](#), “Dlgsnapshot Reference”.

When boards are downloaded and you run Dlgsnapshot manually, a question is posed about running diagnostic firmware. You must confirm that you want Dlgsnapshot to stop the running board and download diagnostic firmware.



Checking Springware Architecture Boards

4

This chapter provides a procedure for testing Springware architecture boards using the UDD diagnostic tool, as well as some related procedures. For more information about the UDD tool, refer to [Chapter 29, “UDD Reference”](#). For information about checking Intel® telecom boards with DM3 architecture, refer to [Chapter 3, “Checking DM3 Architecture Boards”](#). This chapter contains the following sections:

- [Requirements 29](#)
- [Testing Boards with UDD 30](#)
- [Aborting Tests 33](#)
- [Monitoring Test Status 33](#)
- [Ending a UDD Session 34](#)
- [Generating a Log File 34](#)
- [Troubleshooting UDD and Checking Equipment 34](#)

4.1 Requirements

UDD requires the base Intel® Dialogic® System Release Software for your operating system. Refer to the product *Release Guide* for more information.

For Linux only, a Java Runtime Environment is required for operation of UDD.

UDD relies on configuration data used to download firmware to an Intel Dialogic Springware architecture board. Any Intel Dialogic Springware architecture board configured to operate in the system will appear on the initial UDD screen, providing it is supported by the UDD utility. Refer to the *Release Guide* for a list of supported boards.

Note: Before starting the UDD utility, update the Springware configuration files, if necessary, as described in the *Configuration Guide*.

The method used to record configuration data depends on your operating system and the base package installed on your system. Refer to the product *Release Guide* or the *Configuration Guide* for more information about configuration data.

4.2 Testing Boards with UDD

This procedure describes how to test boards using the UDD diagnostics tool. For details about UDD tests and GUI screens, refer to [Chapter 29, “UDD Reference”](#).

1. Stop all active applications on any of the Intel Dialogic Springware architecture boards in the system.
2. Start UDD as described for the operating system you are using:

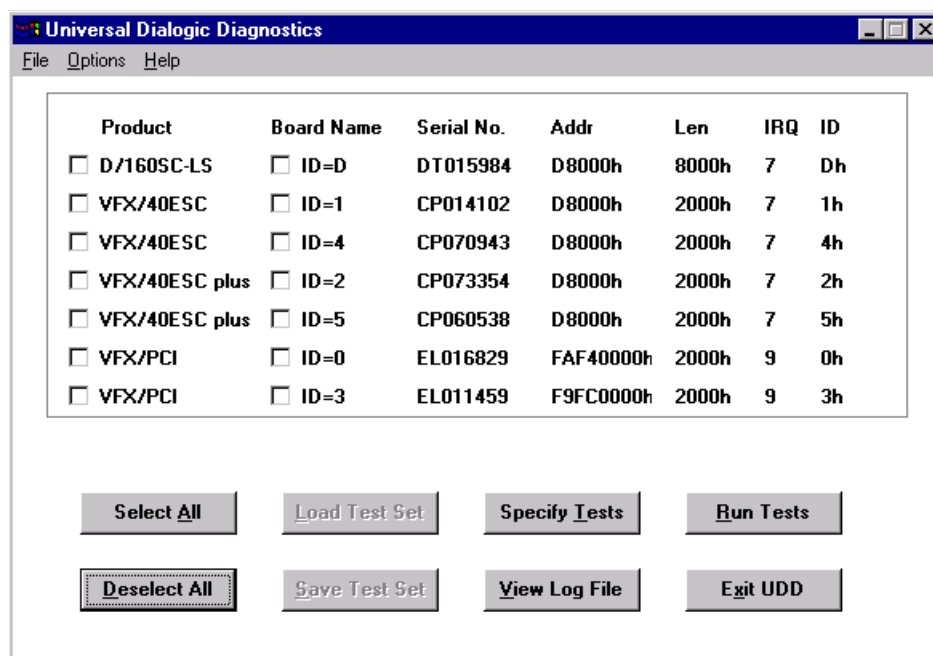
Operating System	Method
Windows	From the Intel Dialogic System Software program folder, click Universal Dialogic Diagnostics Utility .
Linux	From the command line, execute <code>usr/dialogic/bin/udd</code> . This command runs the UDD.

When you start UDD, a Start-up screen with the version and copyright information appears. If error messages appear, refer to [Chapter 29, “UDD Reference”](#) for more information. For Windows, UDD will then display a warning that all Intel Dialogic Springware architecture boards will be stopped. For all operating systems, however, all the Springware boards are stopped so that the UDD test firmware can be downloaded to the boards. For Windows, respond by clicking the **Continue** button on the Warning dialog to continue; otherwise, the UDD utility will exit.

An initialization screen appears next which displays information as the Intel Dialogic Springware architecture boards are found and the test firmware is downloaded. (This information actually comes from the UDD initialization log file, *uddinit.log*. This file is located in `/usr/dialogic/log` for Linux systems; for Windows systems, use the Find Files utility to find the *uddinit.log* file.) You can scroll through this screen if needed. This screen will be covered by the UDD Main screen, but is available for viewing until you click **OK** to close it.

The Main screen appears when the downloading is done. The Main screen appears similar to the one shown in Figure 1.

Figure 1. Example of UDD Main Screen



The Main screen displays all Intel Dialogic Springware architecture boards currently configured in the system. Use the scroll bar, if necessary, to scroll through the complete list of boards. At the bottom of the screen are buttons for selecting screen options. Use the mouse or Tab key to select screen elements.

Note: Text that exceeds the field width will display “. . .” to indicate more text is available. Put the mouse cursor over the field to view the hidden text.

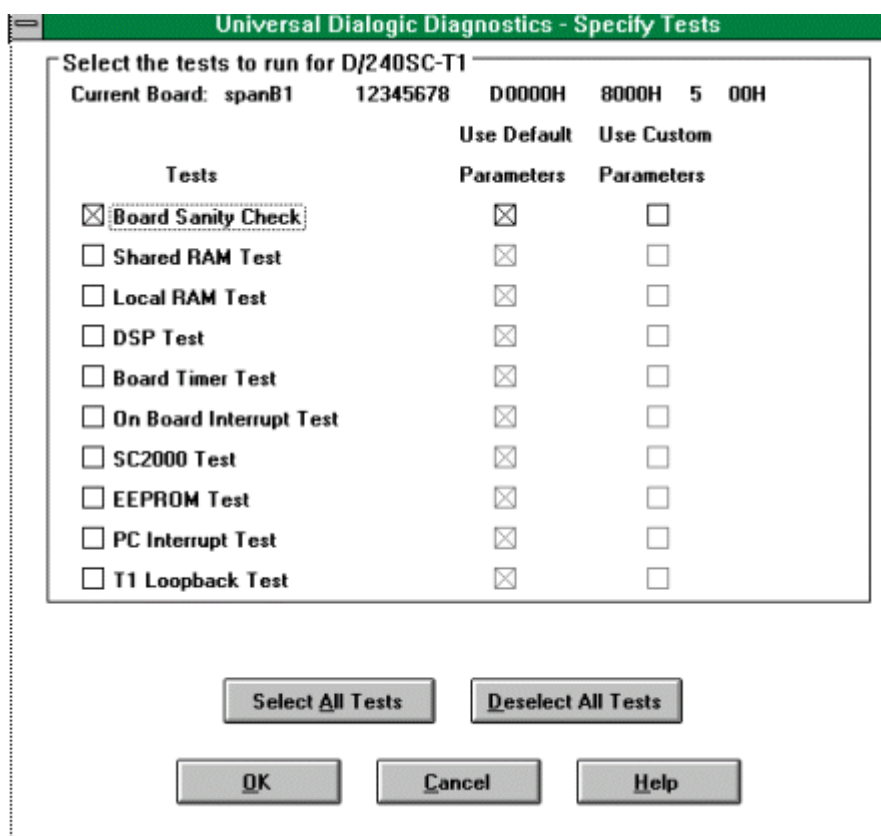
3. Select a board to test by clicking the box next to the name of the board you wish to test.

When a board is selected, a check mark will appear in the box next to the board and the Specify Tests screen will appear. If you want to select more boards, bring the Main screen to the front. Select all of the products and boards you wish to test before you specify the tests to run on the boards.

You may select either a product name or individual board. If you select a product name, each board (daughterboard or dual board) that is part of the product will be selected automatically. If the board was not already selected, a check mark will appear to the left of the product name when you click that box. If the board was previously selected, clicking the box again will deselect it and the check mark will disappear.

4. Click the **Specify Tests** button. A Specify Tests screen (Figure 2) is displayed for each selected board. For a full description of UDD tests and applicable parameters, see [Chapter 29, “UDD Reference”](#).

Figure 2. UDD Specify Tests Screen

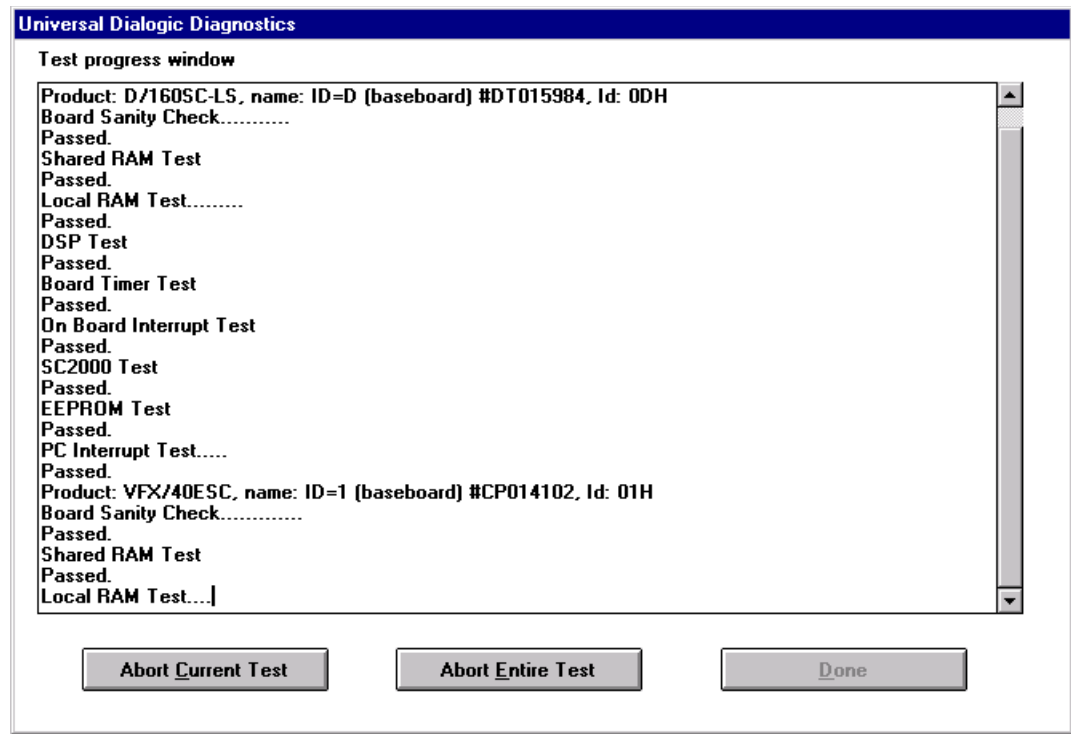


Tests	Use Default Parameters	Use Custom Parameters
<input checked="" type="checkbox"/> Board Sanity Check	<input checked="" type="checkbox"/>	<input type="checkbox"/>
<input type="checkbox"/> Shared RAM Test	<input checked="" type="checkbox"/>	<input type="checkbox"/>
<input type="checkbox"/> Local RAM Test	<input checked="" type="checkbox"/>	<input type="checkbox"/>
<input type="checkbox"/> DSP Test	<input checked="" type="checkbox"/>	<input type="checkbox"/>
<input type="checkbox"/> Board Timer Test	<input checked="" type="checkbox"/>	<input type="checkbox"/>
<input type="checkbox"/> On Board Interrupt Test	<input checked="" type="checkbox"/>	<input type="checkbox"/>
<input type="checkbox"/> SC2000 Test	<input checked="" type="checkbox"/>	<input type="checkbox"/>
<input type="checkbox"/> EEPROM Test	<input checked="" type="checkbox"/>	<input type="checkbox"/>
<input type="checkbox"/> PC Interrupt Test	<input checked="" type="checkbox"/>	<input type="checkbox"/>
<input type="checkbox"/> T1 Loopback Test	<input checked="" type="checkbox"/>	<input type="checkbox"/>

- Specify which test(s) to run by clicking in the box next to a specific test or click **Select All Tests**. To deselect tests, click in the box next to a specific test or click **Deselect All Tests**. To set parameters (such as the number of iterations), click **Use Custom Parameters**.
- After you specify tests for all boards, you must click **OK** so that UDD will return to the Main screen.

As tests run, a Test Progress screen (Figure 3) appears. It indicates the product being tested, the specific test name, and the status of the test.

Figure 3. UDD Test Progress Screen



In Windows, dots across the screen indicate continuing progress. Selecting all tests for multiple boards can take several minutes to run. When all tests are run:

- If successful, the **Done** button is enabled, the **Abort** buttons are disabled and the text “no errors found” appears.
- If any failure occurs, a new screen appears displaying information about the failed board(s). Click the **Details** button on this screen to view the details about the failed board(s).

4.3 Aborting Tests

You may choose to abort a test while a test run is in progress. To abort the current test, click **Abort Current Test** from the test progress screen. To abort all tests, click **Abort Entire Test**. An “Aborting test. Please wait...” message will appear on the screen regardless of the abort option chosen. The final status of the test will be shown as “***ABORTED***”.

4.4 Monitoring Test Status

A test will result in one of the following outcomes:

- PASS indicates successful completion of all iterations.
- FAIL indicates a failure when running a test. (The test ends when the first failure occurs.)

- REJECTED indicates a failure during test initialization.
- ABORTED indicates the user aborted a test.

Upon completion of all tests, a test results screen indicates the product name, test type, and a test result message, for example, FAILED or REJECTED. To receive more information regarding failures, click **Details**.

Note: No details are given for PASS or ABORTED results.

4.5 Ending a UDD Session

When you are done with the UDD session, click **Exit UDD** from the Main screen. You will have to run *dlstart* to download the Intel Dialogic firmware to the boards and start the boards up.

The log file is written over upon restarting UDD. Therefore, if you want to save the results, rename the log file.

4.6 Generating a Log File

A log file, *udd.log*, can be used to store UDD test results. For Windows, the log file is **not** generated by default. To generate a log file of test results, use the mouse to select **Log Test Results** from the **Options** menu off the Main screen toolbar. For Linux, the log file is generated by default, but can be enabled or disabled in the **Options** menu.

This log file is located in `%INTEL_DIALOGIC_DIR%\Bin` for Windows or `${INTEL_DIALOGIC_DIR}/log` for Linux.

4.7 Troubleshooting UDD and Checking Equipment

An installation oversight often creates a problem in an assembled system or testing setup. When isolating a problem, check the physical equipment first. If problems continue, confirm that:

- All boards are firmly seated and secured in their PC expansion slots.
- Daughterboard modules are firmly seated on the baseboard.
- For Linux, confirm that a Java Runtime Environment is installed.
- The IRQ setting of the board you suspect has a problem does not conflict with other expansion boards in your system.
- Each ISA/BLT or hardware configurable board has a unique board locator number or shared RAM address.
- The configuration data is correct.
- The Intel Dialogic System Release Software and SDK are properly installed.
- The UDD software is properly installed.

A known good board can be used to verify that the UDD software has been installed properly.

This chapter describes how to use the PSTN Diagnostics tool (pstndiag) to determine what is preventing your system from successfully completing the first call.

- [Preparation](#) 35
- [Monitoring the Signal](#) 35
- [Monitoring the Status of Alarms](#) 37

For more information about the pstndiag tool, refer to [Chapter 24, “PSTN Diagnostics Tool Reference”](#).

5.1 Preparation

Before you perform the procedures described in this chapter, refer to the following sections:

- [Section 24.2, “Guidelines”](#), on page 100
- [Section 24.3, “Preparing to Run the PSTN Diagnostics Tool”](#), on page 100
- [Section 24.4.1, “Starting the PSTN Diagnostics Tool”](#), on page 100.

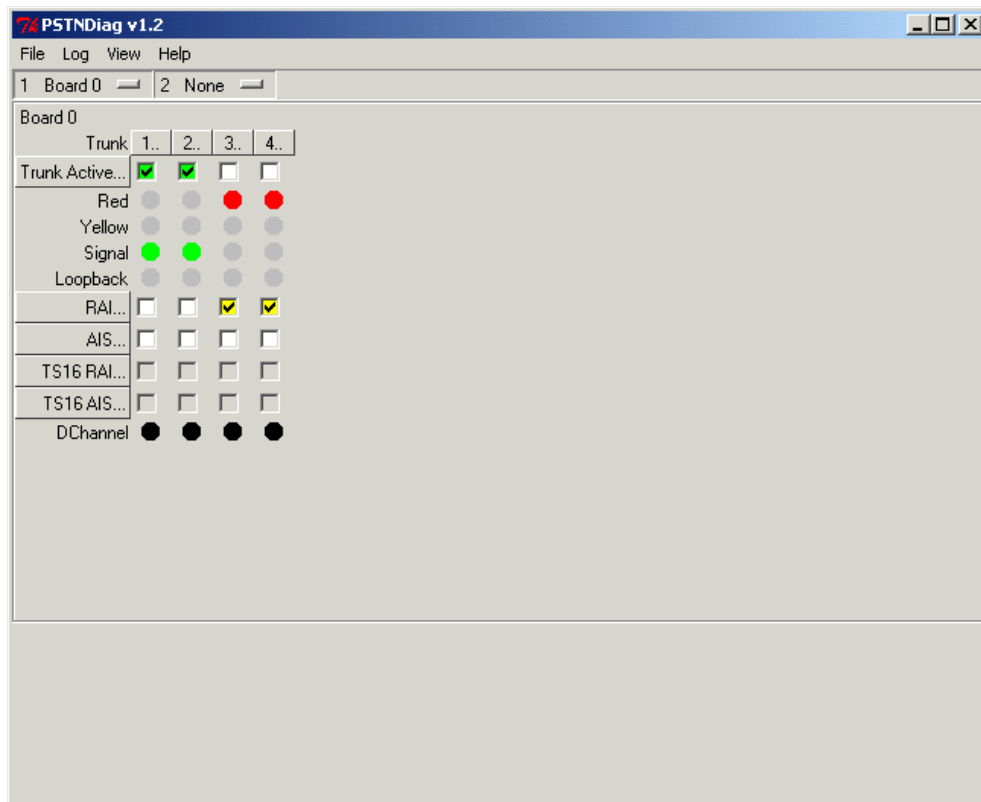
The main window displays a tree or hierarchy of the components in your system. Components include all boards that have been downloaded in the system, all trunks within a board, and all channels within a trunk.

5.2 Monitoring the Signal

Use the following procedure to verify that a connection exists between your board and the network (or other device depending on your system setup):

1. On the pstndiag tool’s main window, double-click on a specific board to display the board level view and check the connection of all trunks on this board. A sample screen is illustrated in [Figure 4](#).

Figure 4. PSTN Diagnostics Tool - Board Level View



- In the board level view, check that the Signal indicator is green for each trunk. The color green indicates a connection between the trunk and another device. The color grey indicates no connection exists. (The various indicators for each trunk, Red, Yellow, Signal, and Loopback, mimic the indicators on the physical board.)
- If the Signal indicator is green for all trunks on the board, then a connection exists between all trunks and another device. You can then proceed to check the status of alarms as described in [Section 5.3, “Monitoring the Status of Alarms”](#), on page 37.

If the Signal indicator is grey for a trunk, no connection exists. The cause may be a hardware problem or a faulty connection.

Note: The Loopback indicator shows whether your board is set for loopback test mode. If loopback is in place, the Loopback indicator is orange; otherwise, the indicator is grey. You should not have your board in loopback test mode for normal operation of the pstndiag tool.

- Repeat steps 1 through 3 for all boards in your system.

5.3 Monitoring the Status of Alarms

After determining that the Signal indicator for all boards in your system is green as described in [Section 5.2, “Monitoring the Signal”](#), on page 35, check the status of alarms on each trunk.

On the pstndiag tool’s main window, double-click on a specific board to display the board level view and check the status of the following alarms on each trunk (see [Figure 4, “PSTN Diagnostics Tool - Board Level View”](#), on page 36 for an illustration):

Red

Red alarm. A red alarm is generated by the line (trunk) of the board being monitored to report a loss of synchronization (RLOS) in the signal being received. This alarm is declared after the condition has existed for a specific time period, typically defined as 2.5 seconds (default). The red alarm condition will exist until the synchronization has been recovered and remains recovered for 12 seconds (default).

Yellow

Yellow alarm. A yellow alarm is generated by the line (trunk) of the board being monitored to signify that a red alarm condition exists at the receiving (local) end. The yellow alarm is sent as long as the red alarm condition exists at the receiver device.

Note: It’s possible for the Signal indicator to be green and a yellow or red alarm to be on at the same time. This situation indicates a possible configuration problem (such as trunk configuration mismatch) or hardware problem.

RAI

Remote Alarm Indication. If on, this box is yellow. This alarm is turned on by the firmware indicating a problem on the remote end. In this situation, the Signal indicator for the trunk will also be red.

AIS

Alarm Indication Service. This alarm can be turned on by you as a test on a specific trunk. This alarm is not typically used.

TS16RAI

TS16 Remote Alarm Indication. This alarm only applies to E1 lines with Channel Associated Signaling (CAS).

TS16AIS

TS16 Alarm Indication Service. This alarm only applies to E1 lines with Channel Associated Signaling (CAS).

D Channel

D channel state. This alarm only applies to ISDN with Common Channel Signaling (CCS). If the indicator is green, this means that the D channel is available. If the indicator is grey, this means that the channel is not available.



This chapter describes how to check the protocol configuration using the PSTN Diagnostics (pstndiag) tool.

- [Preparation](#) 39
- [Checking the Protocol Configuration](#) 39

For more information about the pstndiag tool, refer to [Chapter 24, “PSTN Diagnostics Tool Reference”](#).

6.1 Preparation

Before you use the pstndiag tool as described in this chapter, refer to the following sections:

- [Section 24.2, “Guidelines”](#), on page 100
- [Section 24.3, “Preparing to Run the PSTN Diagnostics Tool”](#), on page 100

After determining that a network connection exists as described in [Chapter 5, “Diagnosing First Call Issues”](#), check the protocol configuration. If the protocol is not properly configured, calls will not be successfully made. To check the protocol configuration, you will make and receive calls on your system. Protocols are configured on a trunk basis.

The procedure described in this section applies to a system with a loopback test setup or a system using a network connection setup.

If you can successfully make and receive calls in a system with loopback test setup, you can rule out any hardware problems. If you then test your system using a network connection setup, and are unsuccessful in making and receiving calls, you will need to check the board configuration settings. The configuration settings on the board and the switch should match; for example, the line type, the line coding, and the signaling protocol.

6.2 Checking the Protocol Configuration

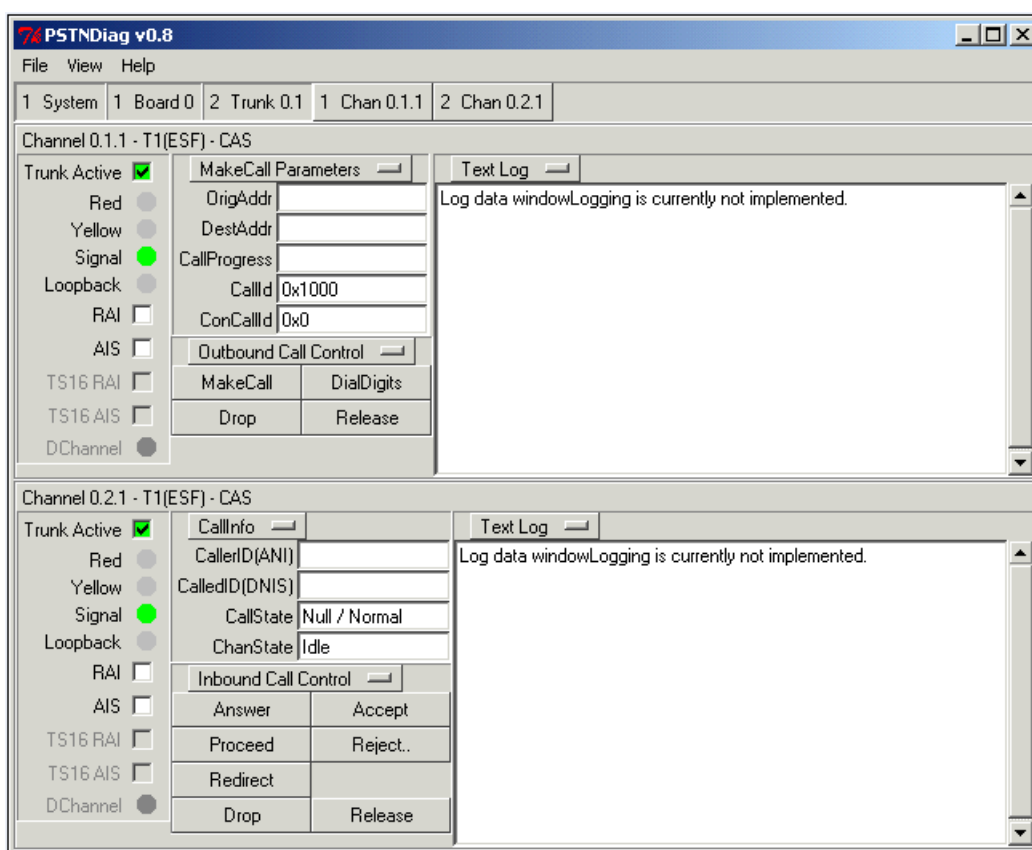
Use the following procedure to check the protocol configuration:

1. If you have not already done so, start the pstndiag tool, as described in [Section 24.4.1, “Starting the PSTN Diagnostics Tool”](#), on page 100.

The main window displays a tree or hierarchy of the components in your system, including all boards that have been downloaded in the system, all trunks within a board, and all channels within a trunk.

2. Shift-double-click on a channel (for example, Chan 0.1.1) to display the channel in the lower pane. This channel will be the transmitting (outbound) channel.
3. Shift-double-click on a channel in another trunk (for example, Chan 0.2.1) to display the channel in the lower pane. This channel will be the receiving (inbound) channel.
4. To view both inbound and outbound channels on your screen, click on the “**1 System**” button (upper pane button) and select “**Chan 0.1.1**” (outbound channel). This channel level view replaces the system level view and is displayed in the upper pane. The inbound channel is displayed in the lower pane.

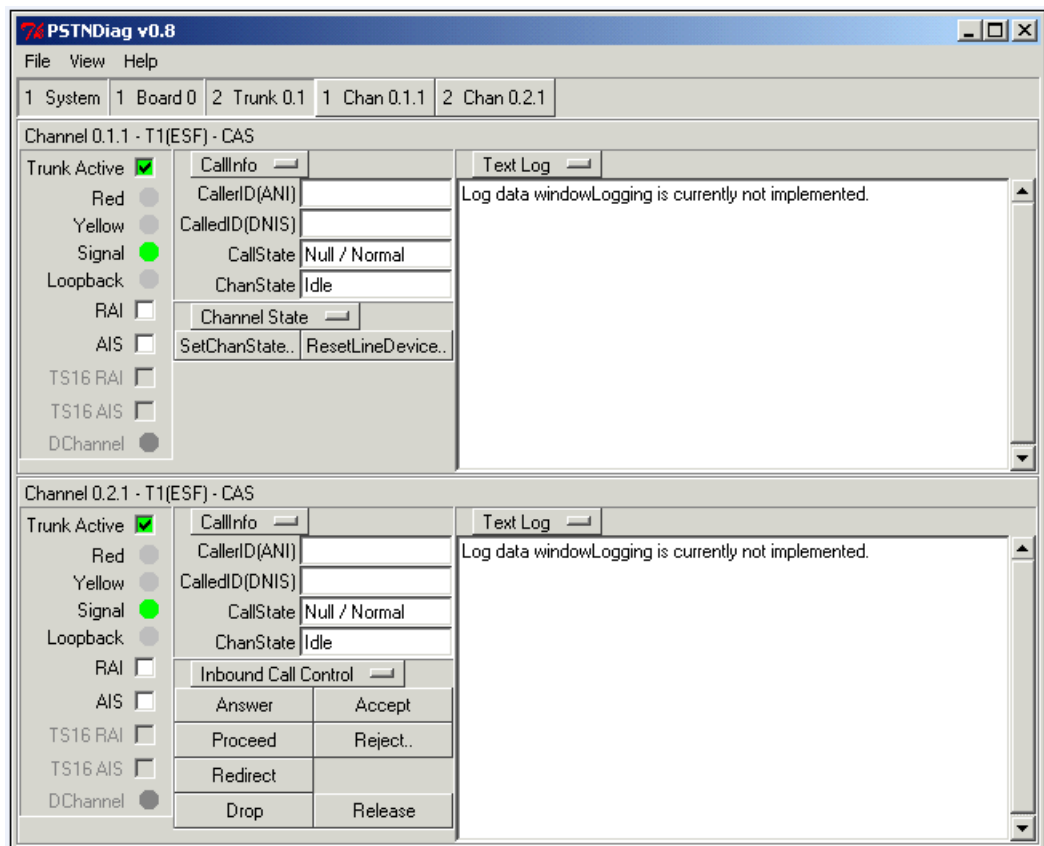
Figure 5. PSTN Diagnostics Tool - Outbound Channel (Upper Pane) & Inbound Channel (Lower Pane)



5. For the outbound channel (upper pane), click the **CallInfo** button to view additional options, and select the **Edit MakeCall Parameters** option. A new set of fields is displayed.
6. Enter the originating telephone number or extension in the OrigAddr field.
7. Enter the destination telephone number or extension in the DestAddr field.

Note: The Call Progress, CallId, and ConCallId fields are not typically used and should not be modified. The Call Progress field specifies the level of call progress analysis on the line. The CallId and ConCallId fields contain internal values used by the firmware to keep track of call information.

Figure 6. PSTN Diagnostics Tool - Making a Call

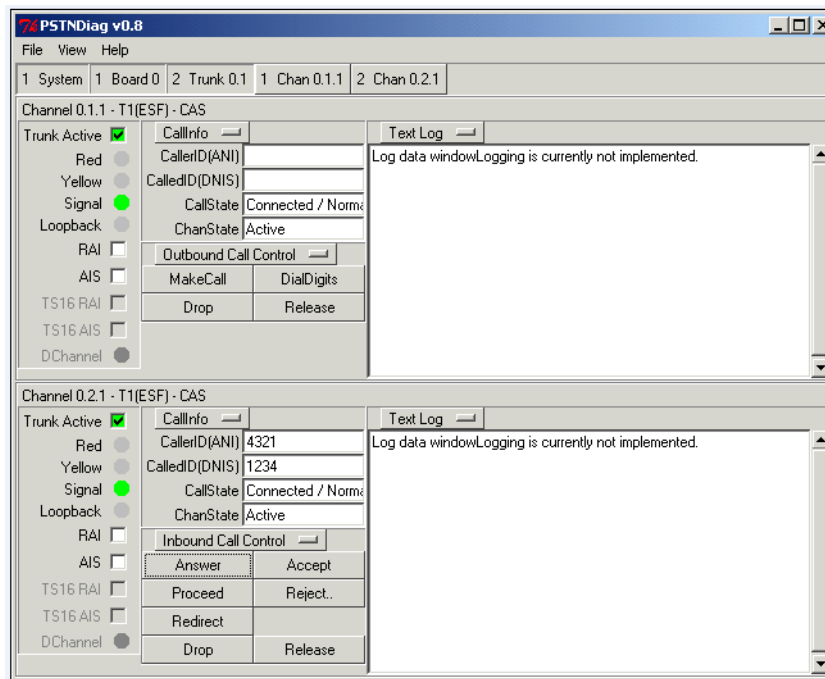


8. For the outbound channel (upper pane), click the **MakeCall Parameters** button to view options, and select the CallInfo button.
9. For the outbound channel (upper pane), click the **SetChanState** button and set the channel in service.
10. For the inbound channel (lower pane), click the **SetChanState** button and set the channel in service.
11. For the outbound channel (upper pane), click the **Channel State** button to view additional options, and select Outbound Call Control Commands.
12. For the inbound channel (lower pane), click the **Channel State** button to view additional options, and select Inbound Call Control Commands.

13. Click the **MakeCall** button to start the call. Notice that the CallInfo fields are updated after a call is made, enabling you to monitor the state of the call. For example, the Called ID (DNIS) field shows the telephone number dialed by a caller. The Caller ID (ANI) field shows the telephone number of the incoming call. Also the CallState field shows the state of the call (such as Connected or not) and the ChanState field shows the state of the channel (such as Active or not).
14. For the inbound channel (lower pane), click the **Accept** button to accept the incoming call. Notice that the CallState for the inbound channel will show “Accepted.” If the expected state is not being displayed, it may indicate a protocol configuration problem.
15. For the inbound channel (lower pane), click the **Answer** button to answer the incoming call. Notice that the CallState for the inbound channel now shows “Connected.” If the expected state is not being displayed, it may indicate a protocol configuration problem.
16. After you have verified that a call is successfully made, click the **Release** button for the outbound channel to end the call. The CallState for the inbound channel will show “Disconnected”. The CallState for the outbound channel will not change yet.
17. Click the **Release** button for the inbound channel to end the call. The CallState for the outbound and inbound channels will show “Null / Normal”. The ChanState for the outbound and inbound channels will show “Idle”. If the expected state is not being displayed, it may indicate a protocol configuration problem.

Note: In a channel level view, a log pane is displayed below the **Text Log** button. This log pane only displays events for that particular channel. To change the view between text format and graphical format, click the **Text Log** button.

Figure 7. PSTN Diagnostics Tool - Call Completed and Released



This chapter provides several procedures that can be useful for general software debugging:

- [Troubleshooting Runtime Issues 45](#)
- [Collecting System Data to Diagnose an Application Failure or Crash 48](#)
- [Setting Automatic Data Collection 49](#)
- [Creating a System Configuration Archive. 49](#)

7.1 Troubleshooting Runtime Issues

This section describes how to use the Runtime Trace Facility (RTF) for tracing the execution path of various runtime libraries. You can use the RTFManager GUI to customize the trace settings or you can manually edit the RTF configuration file. Procedures for each of these approaches are provided in the following sections:

- [Using RTFManager to Customize Trace Settings](#)
- [Manually Editing the RTF Configuration File](#)

Using RTFManager to Customize Trace Settings

You can use the RTFManager GUI to customize and activate the RTF's tracing capabilities as follows:

1. Start RTFManager:
 - On Windows systems, RTFManager can be started from the Start menu. RTFManager is in the program group for the Intel telecom software. RTFManager can also be started by just typing "RTFManager" from the console.
 - On Linux systems, RTFManager can be started by typing `/usr/dialogic/RTFManager` or `RTFManager` if the executable is in the operating system's executable searching path.RTFManager's main window appears.
2. From the **Tools** menu, select **RTF Configuration...** This launches the Runtime Trace Facility Manager Configuration window. The Runtime Trace Facility Manager Configuration window contains the following two tabs:
 - General - The General tab is used to enable/disable RTF tracing and customize the format, readability and location of the log files. Refer to [Section 26.5, "General Tab"](#), on page 116 for more information.

- Filtering - The Filtering tab allows you to determine which libraries to trace and the trace level (Debugging, Program Flow, Warnings, Errors) for each library. Refer to [Section 26.6, “Filtering Tab”](#), on page 118 for information about using the Filtering tab.

Adjust the settings on each of these tabs to suit your needs and click **OK** when you are finished.

3. Start your application. As your application runs, RTF will trace the runtime libraries according to RTF configuration file settings that you created using the RTFManager GUI. Refer to the individual entries in the log file or debug stream to review the trace statements generated by the runtime libraries.
4. If you wish to dynamically edit the RTF trace levels while your application runs, it is not necessary to stop the RTF. Instead, perform the following:
 1. Start RTFManager.
 2. Customize the RTF settings using RTFManager (as described above).
 3. Issue the `rtftool reload` command to reload RTF configuration file and restart the RTF engine.

Note: Keep in mind that changes made to the RTF configuration file will not be reflected in the RTF output until the tool has been reloaded.
5. At any time, you can issue the `rtftool` command to pause/restart RTF. RTF will not provide trace output while it is paused.

For more information about RTFManager, refer to [Chapter 26, “RTFManager Reference”](#). This chapter also tells you how to use RTFManager to customize the log file so that only certain trace statements are displayed (refer to [Section 26.7, “Log File Filtering”](#), on page 120).

Manually Editing the RTF Configuration File

Note: If you run full RTF logging on high-density systems (1000 media channels), you may experience I/O throughput degradation. It is recommended that you do not run RTF with full logging on high-density systems or any field-deployed systems. Instead, use runtime filtering capabilities of the RTF to only log the data you are interested in.

Use the following procedure to manually customize and activate the RTF’s tracing capabilities:

1. Locate the RTF configuration file. The default installation location is determined by the `INTEL_DIALOGIC_CFG` environment variable.

Note: The `INTEL_DIALOGIC_CFG` environment variable is set when the Intel® Dialogic® system software is installed. Refer to the *Intel Dialogic System Release Software Installation Guide* for information about Intel Dialogic system software environment variables. To determine the variable setting on a Linux system, type `echo $INTEL_DIALOGIC_CFG`. To determine the variable setting on a Windows system, type `set INTEL_DIALOGIC_CFG`.
2. If you are familiar with XML syntax, use any ASCII text editor to open the RTF configuration file (*RtfConfigWin.xml* for Windows and *RtfConfigLinux.xml* for Linux). If you are not familiar with XML syntax, open the RTF configuration file with XML editor software.

3. Edit the RTF configuration file to customize the RTF tracing capabilities. Refer to [Section 27.3, “RTF Configuration File”](#), on page 124 for complete information about editing the RTF configuration file. The RTF configuration file includes the following XML tags:
 - **RTFConfig:** This tag’s attributes configure the RTF. All other tags in the RTF configuration file are children of this tag. This tag must appear one time in the RTF configuration file. The RTF tracing capabilities are turned on by default. Refer to [Section 27.3.1, “RTFConfig Tag”](#), on page 125 for complete information about the RTFConfig tag. The RTFConfig tag includes the following child tags:
 - **Logfile:** This is the first child tag of the RTFConfig tag. The Logfile tag’s attributes set the parameters for the RTF logfile output. This tag is an optional part of the RTF configuration file. Refer to [Section 27.3.2, “Logfile Tag”](#), on page 127 for complete information about the Logfile tag.
 - **Global:** This is the second child tag of the RTFConfig tag. The Global tag is used to specify the global configuration. Configuration settings at the global level are valid for all modules in the RTF configuration file. When a module is traced at the global level, all activity related to that particular module is traced. However, global tag settings have the lowest priority; they can be override by settings in individual Module tags. Refer to [Section 27.3.3, “Global Tag”](#), on page 129 for complete information about the Global tag.
 - **Module:** This is the third child tag of the RTFConfig tag. The Module tag is used to specify the trace configuration for a particular module. The default RTF configuration contains pre-defined module tags for traceable runtime libraries. You can edit the **state** attributes of these pre-defined module tags or delete these pre-defined module tags. The default setting for each module’s **state** attribute is 1, meaning that tracing is enabled by default for each module in the *.xml* file. If you choose to delete modules, keep in mind that the *RtfConfigLinux.xml* and *RtfConfigWin.xml* files must contain at least one Module tag. Refer to [Section 27.3.7, “Module Tag”](#), on page 132 for complete information about the Module tag.
4. When you are finished editing the RTF configuration file, save and close the file.
5. Start your application. As your application runs, RTF will trace the runtime libraries according to RTF configuration file settings. Refer to the individual entries in the log file or debug stream to review the trace statements generated by the runtime libraries.
6. If you wish to dynamically edit the RTF trace levels while your application runs, it is not necessary to stop the RTF. Instead, perform the following:
 1. Open the RTF configuration file.
 2. Customize the settings in the RTF configuration file.
 3. Save and close the RTF configuration file.
 4. Issue the `rtftool reload` command to reload RTF configuration file and restart the RTF engine.

Note: Keep in mind that changes made to the RTF configuration file will not be reflected in the RTF output until the tool has been reloaded.
7. At any time, you can issue the `rtftool` command to pause/restart RTF. RTF will not provide trace output while it is paused.

For more information about RTF and manually editing the configuration file, refer to [Chapter 27, “Runtime Trace Facility \(RTF\) Reference”](#).

Note: If you run full RTF logging on high-density systems (1000 media channels), you may experience I/O throughput degradation. It is recommended that you do not run RTF with full logging on high-density systems or any field-deployed systems. Instead, use runtime filtering capabilities of the RTF to only log the data you are interested in.

7.2 Collecting System Data to Diagnose an Application Failure or Crash

This section describes how to use the Intel® Telecom Subsystem Summary Tool (`its_sysinfo`) to collect the system data you will need to send to Intel’s Support Services to troubleshoot an application failure or crash.

Here are two sample scenarios in which you might use the `its_sysinfo` tool to gather system data:

- A telephony application you are running exits or gets terminated unexpectedly. At that point, you would launch the Intel Telecom Subsystem Summary Tool (`its_sysinfo`) to collect the log files and environment information to send to Intel’s Support Services using your credentials.
- Your telephony application doesn't work as expected. For example, an attempt to make a call fails. At that point, you would run the Intel Telecom Subsystem Summary Tool to gather the log files and environment information to send to Intel’s Support Services.

To familiarize yourself with the Intel Telecom Subsystem Summary Tool (`its_sysinfo`) and all the data it collects, refer to [Chapter 20, “Intel Telecom Subsystem Summary Tool Reference”](#).

Follow this procedure to use `its_sysinfo` to collect system data to pass along for troubleshooting:

1. Start the tool.
 - Linux systems – On the command line, enter `its_sysinfo filename` where *filename* is the name you want to give the zip file. The `its_sysinfo` tool will collect system information and compress it into the zip file. If you do not specify any filename, then the information gets compressed in a zip file with the default name *its_sysinfo.zip*.
 - Windows systems:
 1. Click on *its_sysinfo.exe* in `%INTEL_DIALOGIC_DIR%\bin`.
 2. Click the **Generate** button. A dialog box appears on which you must name the archive file into which you want the information to be collected. The default filename is *its_sysinfo.zip*.
 3. Click the **Save** button and `its_sysinfo` will start collecting information.
 4. A pop-up window displaying “Data collection completed. Zip archive was created as <zip file name>” will appear to indicate completion of *its_sysinfo.exe* execution. Click the **OK** button. The archive file is in the location specified in the tool.
2. Use any standard compression/decompression tool (such as WinZip*) to extract and review the data or send the zip file to Intel’s Support Services.

7.3 Setting Automatic Data Collection

The Application Monitor tool will monitor the application you specify and if the application crashes, it will ask you if you want to run the its_sysinfo data collection tool. To do so, follow this procedure:

1. Specify the application you want to monitor as follows:

```
AppMon.com Application Name [arg1] [arg2]
```

AppMon.com is the Windows CLI version of the Application Monitor tool and AppMon.exe is the Windows GUI version. The Linux CLI version is AppMon. There is no GUI version of the tool for Linux.

2. If the application you are monitoring crashes, a prompt will ask if you want to run its_sysinfo. Select “y” or Yes.
3. The CLI versions will prompt for a name for the .zip file in which the data will be collected. The GUI version will launch the its_sysinfo GUI, where you can enter the .zip file name.

You can monitor more than one application at a time by using multiple instances of Application Monitor, but you can only monitor one application per instance of Application Monitor.

Related information:

- If an application has crashed and you did not have it monitored with Application Monitor, you can run its_sysinfo as described in [Section 7.2, “Collecting System Data to Diagnose an Application Failure or Crash”](#), on page 48.
- For more information about how its_sysinfo works and a description of the data it collects, refer to [Chapter 20, “Intel Telecom Subsystem Summary Tool Reference”](#).
- For more information about the Application Monitor tool, refer to [Chapter 8, “Application Monitor Reference”](#).

7.4 Creating a System Configuration Archive

This section describes how to use the Intel® Telecom Subsystem Summary Tool (its_sysinfo) to create a system configuration archive.

As part of a quality control effort, you might want to baseline your systems by retrieving all available information through its_sysinfo. You would archive this baseline system information and use it as a point of comparison to any system that exhibits erroneous behavior. This would provide you with a means of performing an initial cause/effect analysis when you troubleshoot the problem.

To familiarize yourself with the Intel Telecom Subsystem Summary Tool (its_sysinfo) and all the data it collects, refer to [Chapter 20, “Intel Telecom Subsystem Summary Tool Reference”](#).

Follow this procedure to use its_sysinfo to create a system configuration archive:

1. Start the tool.

- Linux systems – On the command line, enter `its_sysinfo filename` where *filename* is the name you want to give the zip file. The `its_sysinfo` tool will collect system information and compress it into the zip file. If you do not specify any filename, then the information gets compressed in a zip file with the default name *its_sysinfo.zip*.
- Windows systems:
 1. Click on *its_sysinfo.exe* in %INTEL_DIALOGIC_DIR%\bin.
 2. Click the **Generate** button. A dialog box appears on which you must name the archive file into which you want the information to be collected. The default filename is *its_sysinfo.zip*.
 3. Click the **Save** button and `its_sysinfo` will start collecting information.
 4. A pop-up window displaying “Data collection completed. Zip archive was created as <zip file name>” will appear to indicate completion of *its_sysinfo.exe* execution. Click the **OK** button. The archive file is in the location specified in the tool.
- 2. Retain the archive file as your baseline system information.

This chapter describes the Application Monitor tool and contains the following sections:

- [Description](#). 51
- [Guidelines](#) 51
- [Options](#). 52

8.1 Description

The Application Monitor tool can be used to monitor an application and if the monitored application terminates abnormally, Application Monitor will launch the `its_sysinfo` tool to collect data that can help in diagnosing the problem. The `its_sysinfo` tool is described in [Chapter 20, “Intel Telecom Subsystem Summary Tool Reference”](#). Procedures for using `its_sysinfo` and Application Monitor can be found in [Chapter 7, “Debugging Software”](#).

The command-line version of the Application Monitor tool is *AppMon.com* on Windows systems and *AppMon* on Linux systems. The GUI version of the Application Monitor tool is *AppMon.exe* and it can be used only on Windows systems.

On Windows systems, Application Monitor is located in `%INTEL_DIALOGIC_DIR%\bin`.

On Linux systems, Application Monitor is located in `${INTEL_DIALOGIC_DIR}/bin`.

8.2 Guidelines

This section contains guidelines for using the Application Monitor tool:

- You can specify one application to monitor for each instance of Application Monitor.
- You can have multiple sessions of Application Monitor running, each monitoring one application.
- If an application terminates abnormally, you will be asked whether you want to run `its_sysinfo`. If you choose to run `its_sysinfo`, you will be prompted to specify a name for the `.zip` file in which the system information is collected. If you choose not to run `its_sysinfo`, the Application Monitor exits.
- When you use the GUI version of Application Monitor (*AppMon.exe*, for Windows only), you must specify the application to monitor in a command line (as described below), but if the application terminates abnormally, a dialog box will appear to ask about running `its_sysinfo`, and if you choose **Yes**, the GUI version of `its_sysinfo` will launch.
- If the application being monitored terminates normally, then Application Monitor will terminate too. When the application being monitored terminates abnormally, then *AppMon* asks if you want to run `its_sysinfo`. If you choose to run `its_sysinfo`, then Application Monitor

terminates after `its_sysinfo` terminates. If you choose not to run `its_sysinfo`, then Application Monitor terminates without running `its_sysinfo`.

8.3 Options

Regardless of whether you use the command line version or GUI version of Application Monitor, you must specify an application to monitor in this format:

Usage:

```
AppMon.com Application Name [arg1] [arg2]
```

Note: The usage is the same for the GUI and command line versions of Application Monitor.

If the application is not in the same directory as Application Monitor, you must specify the full path. For example: `AppMon.com <full path of Application in quotes>`.

If the application being monitored has any arguments, then those can be passed too. For example, if you want to monitor `<program name> <filename>`, then I can execute `AppMon.exe <program name> <filename>`. Here `<filename>` is the argument.

If you use the Windows command line version (*AppMon.com*) or Linux command line version (*AppMon*) and the application terminates abnormally, a message like the following will be output:

```
"Application [ApplicationName] aborted!  
Would you like to run its_sysinfo[y/n]:"
```

If you enter **y**, the following message is output:

```
"Please enter the name of the sysinfo zipped file with the extension  
.zip[default: its_sysinfo.zip]:"
```

After you enter the name of the *.zip* file, `its_sysinfo` is executed and the information is collected in the specified *.zip* file. The Application Monitor application closes after `its_sysinfo` completes.

If you use the GUI version (*AppMon.exe*) and the application terminates abnormally, a message box pops up with the following message:

Process aborted. Do you want to run `its_sysinfo`?

If you click the **Yes** button, the `its_sysinfo` GUI pops up, allowing you to run `its_sysinfo`.

In both of the above cases, if you respond **n** or **No** to the prompt, the Application Monitor exits.

If you want to stop monitoring the application, abort the application by pressing **CTRL-C**. When the application aborts, Application Monitor will ask you if you want to run `its_sysinfo` and you can respond **No**.

This chapter provides reference information about the CallInfo tool. The following topics are included:

- [Description](#). 53
- [Guidelines](#) 53
- [Options](#). 54

9.1 Description

The CallInfo tool detects call information using the DM3 board's Telephony Service Provider (TSP) resource. The CallInfo tool confirms that the TSP component is working correctly by monitoring select messages on a per call basis.

9.2 Guidelines

CallInfo is a QScript utility. For more information about QScript utilities, refer to [Chapter 25, "QScript Reference"](#).

To use CallInfo, specify the call-related messages you want to monitor as follows:

1. Launch the CallInfo tool from the command line.
2. Click the **Action** menu on the CallInfo display. Highlight **Select Ids** and select the group of call-related events you want to trace. The following menu selections are available:
 - CallInfoSet: TSP-related call messages
 - IE Set: Information elements of ISDN-related messages (focuses on small pieces of information)
 - ISDNMsgSet: ISDN-related messages
3. A window of events specific to the event group that you selected will appear. Select the messages you want to trace by clicking on them.
4. After you select messages, click the **Action** menu and choose **DetectEvt**. The tool will start monitoring the messages you selected. As a new call comes in, it will write over the old call information in the CallInfo display. A separate console window will open that tracks messages coming in and shows the message sequence.
5. If you want to stop tracing events you selected and select other events to trace, click the **Action** menu and select **CancelEvt**. Then follow steps 2 through 4 again.
6. To exit the CallInfo tool completely, select **Action > Exit** or close the window.

9.3 Options

The CallInfo tool uses the following command line options:

-board <n>

Logical ID of board (required). Use the listboards utility (Linux) or the Configuration Manager (DCM) (Windows) to obtain the board's logical ID.

Note: The listboards utility is described in the *Administration Guide* for the release and the Configuration Manager is described in the *Configuration Guides* for the (Windows) release.

-line <n>

Line number that the tool will monitor (optional). The default value is 1.

-chan<n>

Channel number that the tool will monitor (optional). The default value is 1.

The following example runs the CallInfo tool on board 0, line 1, channel 1:

```
callinfo -board 0 -line 1 -chan 1
```

This chapter provides reference information about the CAS Trace tool. The following topics are included:

- [Description](#). 55
- [Guidelines](#) 55
- [Options](#). 55

10.1 Description

The CAS Trace tool enables you to track the bit level transitions on a robbed bit or CAS line. CAS Trace prints messages on the screen in real time. The CAS Trace tool can be configured to output the messages into a circular log file.

The CAS Trace tool is used for troubleshooting problems with an entire span. CAS Trace is used for long-term debugging and logging for problems with timing, hung channels, and improper bit states.

10.2 Guidelines

The CAS Trace 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 take several seconds to start the monitoring on all the trunks

The CAS instance is not valid until the line is in service, so you will have to run an application or the Lineadmin and Phone tools to set the channel in-service before using this application. For information about the Lineadmin tool, refer to the Administration Guide for the software release. For information about the Phone tool, refer to [Chapter 23, “Phone Reference”](#).

10.3 Options

The CAS Trace tool uses the following command line options:

-board <board list>

Logical ID of board (or boards) to trace (optional). The default is the lowest ID present in the system. Use the listboards utility (Linux) or the Configuration Manager (DCM) (Windows) to obtain the board's logical ID.

Note: The listboards utility is described in the *Administration Guide* for the release and the Configuration Manager is described in the *Configuration Guides* for the (Windows) release.

-line <list of lines>

Line number that the tool will monitor (optional). The default value is 1.

-k <file size>

Size of each file to create in kilobytes (optional). If this option is not present the file size will be infinite.

-# <number of files to create>

The number of files to trace to in a circular fashion. Each file will be the size specified with -k (optional). The default is a single file.

-f <filename>

This will be the base filename. The board, line, and file numbers will be appended to the end of this name (optional). The default is *CasTrace.log*.

-nodisplay

This flag will turn off the screen output to help reduce CPU overhead (optional). The default is to have the screen display on. The display can also be toggled on and off by pressing *d* during runtime.

-t1 / -e1

This flag will tell the system if you are using a T1 protocol or and E1 protocol (optional). The default is T1. If you specify this incorrectly, you may not be able to initialize and monitor the upper channels.

The following will run the CAS Trace tool on boards 0 and 1 for all four spans. Each line will trace to two 1 MB files:

```
CASrtrace -board 0 1 -line 1 2 3 4 -# 2 -k 1000
```

The output will look like the example below:

```
[      timestamp      ] B## L## T## Rx=ABCD Tx=ABCD +delta(mS)
-----
[10/29 23:33:33.218] B0 L1 T1 Rx=1100 Tx=0000
[10/29 23:33:33.828] B0 L1 T2 Rx=0000 Tx=0000
[10/29 23:33:33.828] B0 L1 T1 Rx=1100 Tx=1100 +610
[10/29 23:33:34.531] B0 L1 T2 Rx=1100 Tx=0000 +703
[10/29 23:33:34.593] B0 L1 T2 Rx=1100 Tx=1100 +62
[10/29 23:33:34.609] B0 L1 T2 Rx=1100 Tx=0000 +16
[10/29 23:33:34.718] B0 L1 T3 Rx=1100 Tx=0000
[10/29 23:33:35.468] B0 L1 T1 Rx=0000 Tx=1100 +1640
[10/29 23:33:35.500] B0 L1 T3 Rx=1100 Tx=1100 +782
[10/29 23:33:35.562] B0 L1 T1 Rx=0000 Tx=0000 +94
[10/29 23:33:35.656] B0 L1 T4 Rx=1100 Tx=0000
[10/29 23:33:35.656] B0 L1 T1 Rx=0000 Tx=0000 +94
[10/29 23:33:36.625] B0 L1 T5 Rx=1100 Tx=0000
[10/29 23:33:36.843] B0 L1 T1 Rx=1100 Tx=0000 +1187
[10/29 23:33:36.906] B0 L1 T1 Rx=1100 Tx=1100 +63
[10/29 23:33:36.921] B0 L1 T1 Rx=1100 Tx=0000 +15
[10/29 23:33:37.203] B0 L1 T3 Rx=0000 Tx=1100 +1703
[10/29 23:33:37.328] B0 L1 T6 Rx=1100 Tx=0000
[10/29 23:33:37.328] B0 L1 T3 Rx=0000 Tx=0000 +125
[10/29 23:33:37.421] B0 L1 T3 Rx=0000 Tx=0000 +93
[10/29 23:33:37.843] B0 L1 T4 Rx=1100 Tx=1100 +2187
[10/29 23:33:37.875] B0 L1 T2 Rx=1100 Tx=1100 +3266
[10/29 23:33:37.875] B0 L1 T5 Rx=1100 Tx=1100 +1250
```


The output is separated into the following columns:

- Timestamp - the time at which the event is received down to msec
- B## - the board's logical ID
- L## - the line number
- T## - the timeslot
- Rx=ABCD - the state of the Receive bits
- Tx=ABCD - the state of the Transmit bits
- Delta - the time between the last transition in msec (if this is blank, it is the initial bit state)

This chapter provides reference information about the DebugAngel tool. The following topics are included:

- [Description](#) 59
- [Guidelines](#) 59
- [Command Line Options](#) 59
- [Additional Configuration Options](#) 60

11.1 Description

The DebugAngel tool provides low-level firmware tracing to aid in low-level debugging. Running as a Windows service or Linux Daemon, it polls the DM3 boards in the system and posts **qPrintf()** statements from the resources and DM3 kernel to a log file.

11.2 Guidelines

Once the service is running, no further action is needed. Any changes to your system (for example, new boards) are automatically detected. On a Windows system, information is logged to the file *DebugAngel.log* in the *%INTEL_DIALOGIC_DIR%\log* directory. On a Linux system, information is logged to the file *debugangel.log* in the *\${INTEL_DIALOGIC_DIR}/log* directory.

11.3 Command Line Options

This section explains the command line options that can be used with the DebugAngel tool.

To install the service and start it running in automatic mode, enter the command:

```
DebugAngel -install
```

The DebugAngel tool uses the following options when it is invoked from the command line.

Notes: 1. Command line options are mutually exclusive.

2. On Linux systems, you must start DebugAngel from *\${INTEL_DIALOGIC_DIR}/bin*.

- instonly
Installs the service without starting it.
- start
Starts the service.
- stop
Stops the service.

- manual
Changes the service startup mode to manual.
- auto
Changes the service startup mode to automatic (default).
- remove
Removes the service (and stops it if it was started).
- status
Shows the service status.

11.4 Additional Configuration Options

Additional configuration options for DebugAngel are available under the Windows registry key *HKEY_LOCAL_MACHINE\SOFTWARE\Dialogic\DebugAngel*.

Warning: Incorrect manipulation of the Windows registry can render your system unusable, requiring that you reinstall Windows. Only a system administrator qualified to modify the registry should change the DebugAngel configuration.

The configuration options in the registry include:

DebugLevel -> 0

1 writes the log information to DebugView.

MaxFileSize -> 0

0 = no max. When the max size is reached, the file is truncated to 0.

LogFile -> default file name

AutoRename -> 1

1 (default) avoids erasing the file when the computer is restarted; the file is renamed (name).bak. Other options are: 0 = don't back up 2 = rename file to (name).(Day/Time_String) 3 = uses multiple files.

MaxFiles -> 1

Used by AutoRename 3 (default is 1 file).

This chapter provides reference information about the Diagbrd tool. The following topics are included:

- Description. 61
- Guidelines 61
- Options. 61

12.1 Description

The Diagbrd tool is used to perform “POST-on-demand” diagnostics. This enables you to locate and replace faulty boards.

- Notes:**
1. The Diagbrd tool can be run on Intel NetStructure® on DM3 architecture boards and Intel NetStructure® IPT Series boards.
 2. Because the DM3post tool now has more functionality than the Diagbrd tool and can be used on both Windows and Linux, the Diagbrd tool will be deprecated in a future release.

12.2 Guidelines

The board must be in a “stopped” state before Diagbrd can be run. If you do not use the reset option (-r), Diagbrd will only retrieve the POST results from the previous reset. If you use the reset option, Diagbrd will reset the specified board, forcing the Control Processor (CP) POST to run. Diagbrd will then retrieve the POST results and provide a PASS/FAIL indication to you. The board will remain in a stopped state following the completion of POST so you must either restart the board (using the startbrd utility) or remove the board (using the removebrd utility).

12.3 Options

The following command line options are used with the Diagbrd tool:

-a<auid>

Specifies a board via its Addressable Unit Identifier (AUID). Use the listboards utility to determine the board's AUID.

Note: The listboards utility is described in the *Administration Guide* for the release.

-p<slot>

Specifies a board via its Thumbwheel. Use the listboards utility to determine the board's Thumbwheel ID.

- b <pcibus> -s <pcislot>
Specifies a board via its PCIBus and PCISlot number. Use the listboards utility to determine the board's PCIBus number and PCISlot number.
- r
Resets the specified board, forcing the Control Processor (CP) POST to run.

Diagnostics Management Console (DMC) Reference

13

This chapter describes the Diagnostics Management Console (DMC) GUI. The following information is included:

- [Description](#) 63
- [Guidelines](#) 64
- [DMC Main Window](#) 64
- [DMC Configuration Dialog](#) 66

Procedures for using the DMC are provided in [Chapter 2, “Using the Diagnostics Management Console \(DMC\)”](#).

13.1 Description

The Diagnostics Management Console (DMC) provides a single portal from which you can launch the diagnostic tools supplied with Intel® telecom software. The DMC also enables you to locate the log files produced by these tools and view them with the appropriate viewer.

The DMC allows you to launch the diagnostic tools remotely through the standard remote control methods provided with the operating system, such as SSH or Remote Desktop.

The DMC performs the following:

- Lists the available diagnostic tools for execution both locally and remotely
- Lists the diagnostic logs available both locally and remotely for viewing
- Launches the diagnostic tools, which execute within their own environment both locally and within the constraints of the integrated remote management facilities of the operating system or an integrated third party provider
- Launches viewers for displaying logged diagnostic data
- Maintains the association of diagnostic log files to appropriate viewers
- Presents error messages when it encounters errors during runtime

Details about using each diagnostic tool are provided in the other chapters of this document. Procedures for using the DMC are provided in [Chapter 2, “Using the Diagnostics Management Console \(DMC\)”](#).

13.2 Guidelines

When you launch the DMC, you will see a list of the available diagnostic tools and log files. The DMC provides an easy means of accessing the tools and log files from one central location, but you must know what you want to do with a given tool or log file and how to use it. Information about diagnostic tasks and tools is provided in this Diagnostics Guide.

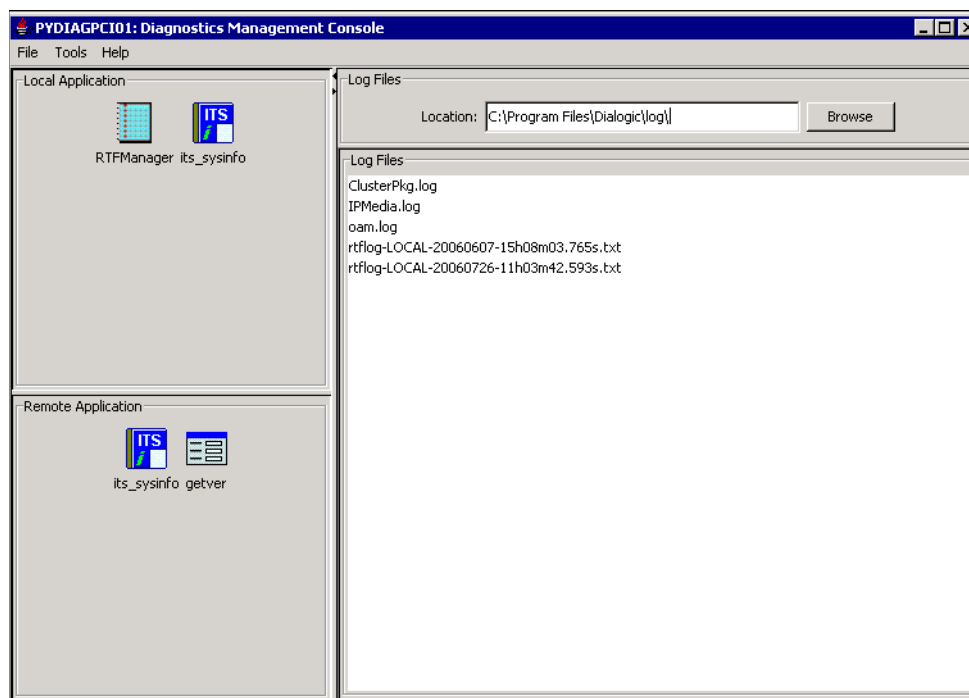
13.3 DMC Main Window

The main window of the Diagnostics Management Console (DMC) is the first window you see after launching the DMC (see Figure 1).

This window consists of the following parts:

- [Menu Bar](#)
- [Diagnostic Application Panels](#) (local and remote)
- [Log File Panel](#)

Figure 1. DMC Main Window



13.3.1 Menu Bar

The main window of the DMC has a menu bar along the top part of the window. This menu bar allows you to access several drop-down menus. Most actions on these menus can also be performed using shortcut keys. The menus, menu items, and shortcut keys are as follows:

File

- **Open log file (CTRL+O)** - Opens the log file that is selected (highlighted) in the log file panel.
- **Execute application (F5)** - Launches the diagnostic tool that is selected (highlighted) in the application panel. For more information about each tool, refer to the tool reference chapters in this Diagnostics Guide.
- **Exit (CTRL+X)** - Closes the DMC.

Tools

- **Configuration (F4)** - Allows you to modify settings for the DMC. For more information, refer to [Section 13.4, “DMC Configuration Dialog”](#), on page 66.

Help

- **Contents (F1)** - Displays online help that describes the DMC and explains how to use the DMC.

13.3.2 Diagnostic Application Panels

On the left side of the Diagnostics Management Console (DMC) main window are two panels: the top panel lists local diagnostic applications and the bottom panel lists remote diagnostic applications. Remote diagnostic applications will be listed only if the DMC has been configured to launch diagnostic applications remotely (see [Section 13.4, “DMC Configuration Dialog”](#), on page 66).

For instructions on launching diagnostic applications, refer to one of the following:

- [Section 2.2, “Launching a Diagnostic Tool”](#), on page 21 (for local machine)
- [Section 2.3, “Launching a Diagnostic Tool on a Remote Machine”](#), on page 22

Note: The terms “diagnostic application” and “diagnostic tool” both refer to the diagnostic tools provided with the Intel telecom software.

13.3.3 Log File Panel

The log file panel is on the right side of the Diagnostics Management Console (DMC) main window. This panel allows you to select a log file to view.

Viewing a log file

To view a log file, locate the file you want in the log file list, click on it to highlight it, and then press **Enter** *or* select **Open log file** from the **File** menu *or* use the **CTRL+O** shortcut. You can also double-click on the log file name to view it. The viewer that is associated with a given log file and located on the local machine will be used to view the log file.

Listing of log files on the local machine

The log file panel of the DMC lists the diagnostic log files available for viewing based on the directory path entered on the [DMC Configuration Dialog](#).

Specifying a different location for log files

You can use the **Browse** button on the log file panel to locate log files. However, if you want to change the default location in which the DMC will look for log files, you can specify it on the [DMC Configuration Dialog](#). You can access the [DMC Configuration Dialog](#) by selecting **Configuration** from the **Tools** menu *or* pressing **F4**. A **Browse** button allows you to select a directory for the Log File Directory field.

Note: If the user-defined variable is empty or not valid, the DMC will use the environment variable set up during installation of the Intel telecom software: INTEL_DIALOGIC_DIR/log.

Listing of log files over the network

The DMC can list log files from mapped drives over the network. You must set this up on the [DMC Configuration Dialog](#). You can access the [DMC Configuration Dialog](#) by selecting **Configuration** from the **Tools** menu *or* pressing **F4**.

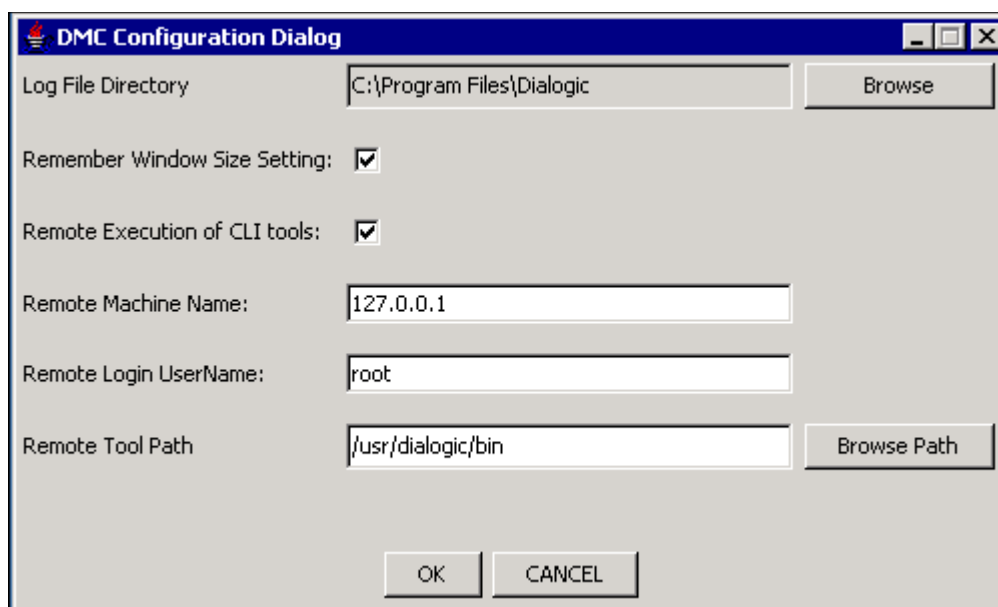
Display of error conditions

Whenever the DMC encounters an error while opening a log file for viewing, it will poll the standard error stream of the operating system for the error message and display it to you.

13.4 DMC Configuration Dialog

The DMC Configuration Dialog (Figure 2) allows you to modify settings for the Diagnostics Management Console (DMC). You must modify these settings if you want to use a diagnostic tool on a remote machine, change the path to the log file directory, or preserve window size settings. You can access the DMC Configuration Dialog by selecting **Configuration** from the **Tools** menu *or* pressing **F4**.

Figure 2. DMC Configuration Dialog



You can modify the following settings using the DMC Configuration Dialog:

- **Log File Directory:** This field shows the location where the DMC will search for log files. If this field is empty or invalid, the DMC will use the environment variable INTEL_DIALOGIC_LOG to specify the log file directory. This field is read-only by default. A **Browse** button allows you to select a directory for this field.
- **Window Size Settings:** A checkbox setting allows you to choose whether you want the DMC to remember its width and height settings since its last run. If you check this box, the next time DMC loads, it will load the window size from its previous execution. If you do not check this box, the DMC will load using the default value for width and height: 800 X 600 pixels.
- **Remote Execution of CLI Tools:** A checkbox setting allows you to use the DMC to execute diagnostic tools remotely via Command Line Interface (CLI). If you check this box, you will see (on the main screen) a list of all the diagnostic tools available to execute via CLI from a remote machine. If you do not check this box, you will not see a list of diagnostic tools available for remote use and you will not be able to use the diagnostic tools remotely.
- **Remote Machine Name:** If you want to use diagnostics tools on a remote machine, you must enter the name of the remote machine here. The DMC will pass on this machine name to remote methods such as SSH to connect to remote machines. If this field is not empty, the DMC will try to use the machine name with remote tools. If the field is empty, when you attempt remote execution the DMC will launch a dialog box in which you can provide the machine name. In either case, if the machine name is invalid, the DMC will show an error and discontinue execution of the remote tool.
- **Remote Login UserName:** You must enter the user name used for logging into the remote machine.
- **Remote Tool Path:** You must specify the path of the diagnostic tool that you will use remotely via a CLI. You can use the **Browse Path** button to specify the path.



The DMC configuration settings are stored in a *dmc.properties* file even when the DMC is shut down. When you start the DMC, it will reload the settings from this file. If this file is not found or cannot be read for any reason, the DMC will use default values hard coded in the application. If there is an error in saving the file for any reason (for example, unavailable disk space or I/O errors of any kind), the DMC will notify you and abort saving the values.

This chapter provides reference information about the DigitDetector tool. The following topics are included:

- [Description](#). 69
- [Guidelines](#) 69
- [Options](#). 69

14.1 Description

The DigitDetector tool demonstrates the use of a DM3 board's Tone Generator and Signal Detector components. It provides the ability to detect digits at the local end of a channel connection.

14.2 Guidelines

DigitDetector is a QScript utility. For more information about QScript utilities, refer to [Chapter 25, "QScript Reference"](#).

To use the DigitDetector tool, you need a physical connection to the board. For example, two network interface trunks can be looped or you might use a connection between end users. You can then use the Phone tool at one end of the channel to generate dialed digits that can be detected by the Digit Detector tool. For information about the Phone tool, refer to [Chapter 23, "Phone Reference"](#).

14.3 Options

The DigitDetector tool uses the following command line options:

-board <n>

Logical ID of board (required). Use the listboards utility (Linux) or the Configuration Manager (DCM) (Windows) to obtain the board's logical ID.

Note: The listboards utility is described in the *Administration Guide* for the release and the Configuration Manager is described in the *Configuration Guides* for the (Windows) release.

-line <n>

Line number that the tool will monitor for digits (optional). The default value is 1.

-chan<n>

Channel number that the tool will monitor for digits (optional). The default value is 1.

The following example runs the DigitDetector tool on board 0, line 1, channel 1:

```
digitdetector -board 0 -line 1 -chan 1
```

This chapter provides reference information about the Dlgsnapshot tool. The following topics are included:

- [Description](#) 71
- [Guidelines](#) 71
- [Options](#) 72

15.1 Description

The Dlgsnapshot tool uses Intel® Dialogic® system software fault monitoring components to generate a core dump file when one of the following faults is detected on a DM3 board:

- Control Processor (CP)
- Signal Processor (SP)

The autodump feature of the Dlgsnapshot tool is disabled by default. However, you can enable 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.

If autodump is enabled, when a fault is detected, a core dump file is created in the *log* directory under *INTEL_DIALOGIC_DIR* and the board that the fault occurred on is taken out of service (stopped) without causing any other interruptions to the system. To reactivate the out-of-service board, you must restart it.

You can also run the Dlgsnapshot tool on demand from the command line.

Note: Dlgsnapshot is for use only on Intel® Dialogic® or Intel NetStructure® boards with a DM3 architecture. However, the SRAM dump is not supported on The Intel NetStructure® DMV-B, DMN160TEC, and DMT160TEC boards.

15.2 Guidelines

Windows systems: When a core dump file is generated, you can send a *.zip* file containing the contents of the *%INTEL_DIALOGIC_DIR%log* directory along with the Windows event viewer system log file (**.evt*) to Intel's telecom support resources for debugging purposes.

Linux systems: When a core dump file is generated, you can send a *.tar* file containing the contents of the *\${INTEL_DIALOGIC_DIR}/log* directory to Intel's telecom support resources for debugging purposes.

When boards are downloaded and you run Dlgsnapshot manually, a question is posed about running diagnostic firmware. You must confirm that you want Dlgsnapshot to stop the running board and download diagnostic firmware.

Each core dump file is named according to the type of fault detected and the date/time the fault occurred. The naming convention of the core dump files is

`faulttype_MM_DD_YY HH_NNxx_MS`

where:

- *faulttype* denotes the type of fault (CPDump, SPDump, SRAM, Counter [Windows only], or Driver State)
- *MM* is a two-digit number indicating the month
- *DD* is a two-digit number indicating the day
- *YY* is a two-digit number indicating the year
- *HH* is a two-digit number indicating the hour
- *NN* is a two-digit number indicating the minute
- *xx* indicates whether the fault occurred in the AM or PM
- *MS* is a number indicating the milliseconds

Note: If multiple DSPs on the same board generate a fault, the core dump file will only contain information about the DSP that generated the initial fault.

Following are some examples of file names:

- CounterDump_11_03_2003 03_13PM_35
- DumpSram_11_03_2003 03_13PM_834
- StateDump_11_03_2003 03_13PM_35

All the dump files have a **.txt* extension except the SRAM dump in Windows, which has a **.bin* extension.

15.3 Options

This section contains the following information:

- [Enabling Autodump](#)
- [Running Dlgsnapshot on Demand](#)
- [Command Line Options](#)

15.3.1 Enabling Autodump

If you want to enable autodump, open the *dlgproductagent.cfg* file located in the *cfg* directory and uncomment out the following line:

```
DIAGNOSTIC_AGENT      = libdlgdm3diagagent
```


The default looks like this (the line you must change is in bold):

```
[DM3]
;DETECTOR_AGENT      = libdlgdm3detectoragent
FAULTDETECTOR_AGENT = libdlgdm3faultdetectoragent
;INITIALIZER_AGENT   = libdlgdm3initializeragent
;CLOCKING_AGENT       = libdlgdm3clockagent
;DIAGNOSTIC_AGENT     = libdlgdm3diagagent
```

You must change it to the following (bold text shows the changed line):

```
[DM3]
;DETECTOR_AGENT      = libdlgdm3detectoragent
FAULTDETECTOR_AGENT = libdlgdm3faultdetectoragent
;INITIALIZER_AGENT   = libdlgdm3initializeragent
;CLOCKING_AGENT       = libdlgdm3clockagent
DIAGNOSTIC_AGENT     = libdlgdm3diagagent
```

When you make this change, Dlgsnapshot will run automatically when a fault is detected.

15.3.2 Running Dlgsnapshot on Demand

The Dlgsnapshot tool can also be run on demand from the command line. See [Section 15.3.3, “Command Line Options”](#), on page 73. Running Dlgsnapshot from the command line may also stop the board, depending on the options selected.

If you try to run the Dlgsnapshot tool on a board version that doesn't support it, you will get a message that Dlgsnapshot is not supported and the command failed.

15.3.3 Command Line Options

The Dlgsnapshot tool uses the following command line options:

-f<number 1 to 6>

This option, which is **mandatory**, indicates which print buffer will be dumped. Values are as follows:

- 1 – SRAM (not applicable to all releases)
- 2 – DRIVER_BOARD_STATE_DUMP
- 3 – DRIVER_BOARD_COUNTER_DUMP (Windows only)
- 4 – DOWNLOAD_OFFDIAG
- 5 – PROCESSOR_DUMP_ONLY (this value requires that you use the **-r** option)
- 6 – DUMP_ALL (this value requires that you use the **-r** option)

-a<AUID>

Dumps the print buffer of the processor specified with the **-r** option on the board with the given Addressable Unit Identifier (AUID).

-r<processor number>

The **-r** option is mandatory with **-f** option 5 or 6.

-l<logical ID>

Dumps the print buffer of the processor specified with the **-r** option on the board with the given logical ID.

-p<physical slot number>

Dumps the print buffer of the processor specified with the -r option on the board with the given physical bus number.

-b<PCI bus number> -s<PCI slot number>

Dumps the print buffer of the processor with the -r option on the board with the given PCI bus number and given PCI slot number.

-h

Displays the help screen.

-v

Displays the version number of Dlgsnapshot.

The following shows the usage of the command and its options:

```
dlgsnapshot [-a -r -f] [-l -r -f] [-p -r -f] [-b -s -r -f] [-h] [-v]
```

Example: To dump all for a board with AUID=50002 r1 (CP), you would use the following:

```
dlgsnapshot -a50002 -r1 -f6
```

Example: To dump all for a board with AUID=50002 r2 (SP2), you would use the following:

```
dlgsnapshot -a50002 -r2 -f5
```

This chapter provides reference information about the DM3post tool. The following topics are included:

- [Description](#) 75
- [Guidelines](#) 75
- [Options](#) 76

A procedure for using this tool is provided in [Section 3.2, “Checking an Individual Board”](#), on page 25.

16.1 Description

The DM3post tool, sometimes referred to as “POST-on-demand”, can perform diagnostics on a stopped board at any time to detect and isolate possible hardware faults.

Note: This tool can be run on Intel NetStructure® on DM3 architecture boards and Intel NetStructure® IPT Series boards.

16.2 Guidelines

The board must be in a “stopped” state before the DM3post tool can be run. If you do not use the reset option (-r), DM3post will *not* reset the board and will only retrieve the POST results for the last reset that occurred. If you use DM3post with the reset option, DM3post will force a full reset of the specified board, forcing the Control Processor (CP) POST diagnostics to run. DM3post will then retrieve the POST results from the SRAM and provide a PASS/FAIL indication to you. The board will remain in a stopped state, so you must restart the board.

DM3post also provides an option to run POST on a chassis level. By using the chassis option (-c), DM3post will retrieve the results of the last run POST for all DM3 boards in the chassis. By using the chassis option with the reset option, you can run POST on all DM3 boards in the system. When using the chassis option, it is not necessary to provide the bus and slot numbers. Any option other than the reset option will be ignored when using the chassis option. In addition to output on the screen, more detailed output is logged to a log file, *dm3post.log*, by default.

- Notes:**
1. Signal Processors (SP) are not diagnosed by DM3post. However, SP health is verified as part of the board’s download process. Therefore sufficient diagnostic coverage for hardware is obtained when a board passes POST and is successfully downloaded.
 2. On Windows systems, for the slot number (-s option), provide the PCI slot number. This information may be obtained from the Configuration Manager (DCM), which is described in the Configuration Guides for the release. On Linux systems, for the slot number, provide the physical slot number. This information may be obtained from the listboards utility, which is described in the *Administration Guide* for the release.

16.3 Options

The following command line options are used with the DM3post tool:

-s<n>

slot number (required). On Windows systems, use the Configuration Manager (DCM) to obtain the board's slot number. On Linux systems, use the listboards utility, which is described in the *Administration Guide* for the software release.

Note: The listboards utility is described in the *Administration Guide* for the release and the Configuration Manager is described in the *Configuration Guides* for the (Windows) release.

-b<n>

bus number (optional if there is only one bus or if the slot number is unique)

-l

logs event (optional). Output is logged to *dm3post.log*.

-q

suppresses output (optional). The tool operates in silent mode.

-r

resets board before retrieving diagnostics results (optional). If not set, results displayed will be those generated at board startup.

-h

displays the tool's help screen

-v

displays the program version

Example 1: Run DM3post on a board in slot 17, bus 0

The following example runs DM3post on a board in slot 17, bus 0:

```
dm3post -s17 -b0
```

You will get the following response:

```
You have chosen to read the initial POST diagnostic results from the board in slot 17, bus 0. No  
board reset will occur.
```

```
Do you wish to continue (y/n)?
```

If you answer Y, the following will be printed to the screen:

```
Retrieving results...
```

The success/failure message will be printed to the screen when POST is complete. Here is an example:

```
SUCCESS: POST passed for board in slot 17, bus 0. Diagnostic Codes: 0x03 0xfc
```

Example 2: Dm3 post run with the reset option on a board in slot 17, bus 0

The following example runs DM3post with the reset option on a board in slot 17, bus 0:

```
dm3post -s17 -b0 -r
```

You will get the following response:

```
You have chosen to run diagnostics on the board in slot 17, bus 0.
```

```
Do you wish to continue (y/n)?
```

If you answer Y, the following will be printed to the screen:

```
dm3post processing...
```

The success/failure message will be printed to the screen when POST is complete. Here is an example:

```
SUCCESS: POST passed for board in slot 17, bus 0. Diagnostic Codes: 0x03 0xfc
```


This chapter provides reference information about the DM3Trace tool. The following topics are included:

- Description. 79
- Guidelines 79
- Options. 79

17.1 Description

The DM3Trace tool acts as a virtual serial port (terminal emulation) session to Intel NetStructure® DM3 architecture boards. This program continuously prints out the qTrace statements included in the DM3 kernel with the **qTraceLevel()** function. This program polls the board and posts the qTrace statements from the resources and DM3 kernel to the screen and designated output file.

17.2 Guidelines

Dm3trace acts as a virtual serial port (terminal emulation) session to the DM3 board, and has the following characteristics:

- If the firmware prints out a string using the **qTrace()** function call, the output string goes to dm3Trace.
- Dm3Trace requires both a board number and a processor number.
- Dm3Trace posts **qTrace()** from the indicated processor.
- Dm3Trace will filter out the output of **qTrace()** that is not in the current trace level.

17.3 Options

The DM3Trace tool uses the following command line options:

-b<n>

Logical ID of board (required). Use the listboards utility (Linux) or the Intel® Dialogic® Configuration Manager (Windows) to obtain the board's logical ID.

Note: The listboards utility is described in the *Administration Guide* for the release and the Configuration Manager is described in the *Configuration Guides* for the (Windows) release.

--d<level>

Do not modify. Leave this at the default value of 0.

- f<file>
Output file name (required to save output in a file).
- p<n>
Processor number (required). Lowest allowable value is 1.
- t<n>
Reserved value. Leave this at the default value of 0.
- h
Displays the help screen.
- v
Displays the version number.

The following example runs the DM3Trace tool on board 1, processor 1 and displays output to the screen:

```
dm3trace -b1 -p1
```


This chapter provides reference information about the getver tool.

- Description. 81
- Options. 81
- Output 82

18.1 Description

The getver (Get Version) software application outputs version information for files that are part of the Intel® Telecom software installation. You specify the file for which you want the version. For example, the following command prints the version string of the *ssp.mlm* file to the screen:

```
getver ssp.mlm
```

Using getver, you can get version information for the following files provided that they follow supported versioning formats:

- Intel NetStructure® DM3 architecture board firmware files: *.srec, *.mlm
- Intel NetStructure DM3 architecture board files: *.hot, *.psi
- Windows* Dynamic Load Library files: *.dll
- Linux* Dynamic Load Library files: *.so
- Other files part of the Intel Telecom software that follow supported versioning formats (the OA&M executable binaries and libraries have a version that getver can display)

Getver is located in the *bin* directory:

- Windows: %INTEL_DIALOGIC_DIR%\bin
- Linux: \${INTEL_DIALOGIC_DIR}/bin

18.2 Options

This section provides the command line arguments for getver:

```
getver -? <filename> -s -CPU <argument value>
```

Note the following:

- If you are using switches or options, you must use the sequence given above.
- -? is a option argument that prints out usage information
- -s acts as a switch for processing *srec* files, to be used only if the *srec* file does not end with the *.srec* extension.

- -CPU is a case-sensitive option argument to be used for *srec* files only if *getver* <filename> fails. Some possible values are PPC, ONYX, and C6X.

Note: You must specify the path to the file if you do not execute *getver* from the directory in which the file is located (in this case, the *data* directory).

18.3 Output

Getver will print the version string of the specified file to the screen unless it encounters an unknown versioning format. In such a case, *getver* will print the following message to the screen:

```
Version format not recognized - file not processed
```

Here are some examples of output:

- Expected output for new “DLcid” versioning format:
Version: <The version number string following "DLcid">
- Expected Output for old “DLcid” versioning format
Embedded name: <embedded name>
Version: <version number> Build <build number>
- Expected output for Windows versioning format:
Version: <version number>

This chapter provides reference information about the ISDNtrace tool. The following topics are included:

- [Description](#) 83
- [Guidelines](#) 83
- [Options](#) 83

19.1 Description

The ISDNtrace tool provides the ability to track Layer 3 (Q.931) messages on the ISDN D-channel. ISDNtrace prints messages on the screen in real time. This trace information can also be captured into a file.

19.2 Guidelines

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. An exception may also be generated if you use an invalid parameter in the command.

19.3 Options

Options

The ISDNtrace tool uses the following command line options:

-b<n>

Logical ID of board (required). Use the listboards utility (Linux) or the Configuration Manager (DCM) (Windows) to obtain the board's logical ID.

Note: The listboards utility is described in the *Administration Guide* for the release and the Configuration Manager is described in the *Configuration Guides* for the (Windows) release.

-d<n>

The D-channel number (trunk number) on the specified board (required). The default value is 1.

-f<file>

Output file name (required to save output in a file).

Note: A space is used after the -f option but not after -b or -d options.

The following example runs the ISDNtrace tool on board 0, D-channel 1 and prints the output to a *trace.txt* file:

```
isdntrace -b0 -d1 -f trace.txt
```

Intel Telecom Subsystem Summary Tool Reference

20

This chapter describes the Intel® Telecom Subsystem Summary Tool (*its_sysinfo*) and the information it collects. This chapter contains the following information about *its_sysinfo*:

- Description. 85
- Command Line Interface. 85
- Graphical User Interface (Windows only). 86
- Information Collected by *its_sysinfo*. 86
- System Information Data Structuring 88

20.1 Description

The Intel Telecom Subsystem Summary Tool (*its_sysinfo*) provides a simple way to collect information about systems built using Intel® telecom products. 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 *its_sysinfo* tool also enables you to collect baseline information about the system for quick review of configuration issues when determining system configuration consistency. This information is collected in a file, compressed, and archived as part of the complete system information collection in an archive file named *its_sysinfo.zip* (or a name you specify). If the installed system is configured in such a way that the baseline information is not available, the *its_sysinfo* tool will indicate “No Information Available.”

In addition to the standard information captured by the *its_sysinfo* tool, you can manually add files to the archive that is created by *its_sysinfo.exe* after you run the tool so you can preserve additional information that might help with resolving issues.

Procedures for using this tool can be found in the following sections:

- Section 7.2, “Collecting System Data to Diagnose an Application Failure or Crash”, on page 48
- Section 7.4, “Creating a System Configuration Archive”, on page 49

20.2 Command Line Interface

On the command line, enter *its_sysinfo filename* where *filename* is the name you want to give the zip file (it can be the complete path). The *its_sysinfo* tool will collect system information

and compress it into the zip file. If you do not specify any filename, then the information gets compressed in a zip file with the default name *its_sysinfo.zip*.

20.3 Graphical User Interface (Windows only)

On a Windows system, you can use a GUI to run *its_sysinfo* as follows:

1. Click on *its_sysinfo.exe* in %INTEL_DIALOGIC_DIR%\bin.
2. Click the **Generate** button. A dialog box appears on which you must name the archive file into which you want the information to be collected. The default filename is *its_sysinfo.zip*.
3. Click the **Save** button and *its_sysinfo* will start collecting information.
4. A pop-up window displaying “Data collection completed. Zip archive was created as <zip file name>” will appear to indicate completion of *its_sysinfo.exe* execution. Click the **OK** button. The archive file is in the location specified in the tool.

20.4 Information Collected by *its_sysinfo*

The following information is collected under *its_sysinfo.htm*, which is one of the files that is added to the archive:

- **General System Information**
 - **Environment Variables** – information about the operating system environment variables
 - **System Event Logs** – See **Table 1**.
 - **Memory and Processor** – information about the platform’s available and used memory and CPU type and number as of the time you executed the *its_sysinfo* tool
 - **Operating System** – information about the operating system’s version, service pack, and language
 - **/proc/meminfo file** – Linux only
- **Information Specific to Intel Telecom Products**
 - **Installed Devices** – *its_sysinfo* collects information about devices detected in Intel Telecom Subsystem configurations and startup.
 - **Configuration Settings for Installed Devices** – *its_sysinfo* captures the stored values used by the configuration tool for Intel Telecom Subsystem configurations and startup. For more information about the configuration tool, refer to the configuration guide(s) for the system release.
 - **Firmware Files and Versions** – *its_sysinfo* collects information about the files listing all firmware file names and version numbers.
 - **FCD Files** – *its_sysinfo* collects information about the *.fcd* files used by the configuration tool for Intel Telecom Subsystem configurations and startup. For more information about FCD files, refer to the configuration guide(s) for the system release.
 - **PCD Files** – *its_sysinfo* collected information for the *.pcd* file used by the configuration tool for Intel Telecom Subsystem configurations and startup. For more information about PCD files, refer to the configuration guide(s) for the system release.
 - **CONFIG Files** – *its_sysinfo* collects information about the *.config* file used by the configuration tool for Intel Telecom Subsystem configurations and startup. For more

information about CONFIG files, refer to the configuration guide(s) for the system release.

- **Global Call Configuration** – its_sysinfo collects information about the Global Call PDK subsystem configuration, which is contained in the *pdk.cfg* file. This file specifies the Global Call protocol modules and the country dependent parameter settings downloaded to each device. For more information about Global Call Configuration, refer to the *Global Call Country Dependent Parameters (CDP) Configuration Guide*, which can be found on this website: <http://resource.intel.com/telecom/support/releases/protocols/index.htm>.
- **Build Information** – its_sysinfo collects information about the contents of the buildinfo.ini file used for the installed Intel Telecom Software Subsystem. The *buildinfo.ini* file contains information about what is in the build of the software.
- **Board Memory Dumps**
- **Intel Telecom Subsystem Event Viewer Data** – its_sysinfo collects information about the last events reported to the operating system and to the event logger from the Intel Telecom Subsystem.
- **File Versions**

The its_sysinfo tool also checks for the log files listed in Table 1 and adds them to the archive.

Note: It is possible to specify a name for some log files. However, its_sysinfo only collects files with default names.

Table 1. Log Files Archived by its_sysinfo

Log Files	Windows	Linux
OA& M Files	\$(SystemRoot)\System32\anm_debug.log \$(SystemRoot)\System32\anm_trace.log %INTEL_DIALOGIC_DIR%\log\ClusterPkg.log %INTEL_DIALOGIC_DIR%\log\ClusterPkg.log.0 Or \$(SystemRoot)\System32\ClusterPkg.log* \$(SystemRoot)\System32\confslot.log %INTEL_DIALOGIC_DIR%\log\ctbb*.log Or \$(SystemRoot)\System32\ctbb*.log dlgsInstall.log is in \$(TEMP) %INTEL_DIALOGIC_DIR%\log\dlgsyslogger.log %INTEL_DIALOGIC_DIR%\log\DM3AutoDump.log \$(SystemRoot)\System32\dm3bsp.log \$(SystemRoot)\System32\dm3fdspdll.log \$(SystemRoot)\System32\frustatus.log %INTEL_DIALOGIC_DIR%\log\GenLoad.log %INTEL_DIALOGIC_DIR%\log\merc.log Or \$(SystemRoot)\System32\merc.log %INTEL_DIALOGIC_DIR%\log\ncm.ini %INTEL_DIALOGIC_DIR%\log\oam.log %INTEL_DIALOGIC_DIR%\log\Sctsassi.log	\$(INTEL_DIALOGIC_DIR)/log/board*.log \$(INTEL_DIALOGIC_DIR)/log/clusterpkg.log \$(INTEL_DIALOGIC_DIR)/log/clusterpkg.log.* \$(INTEL_DIALOGIC_DIR)/log/dlgsyslogger.log \$(INTEL_DIALOGIC_DIR)/log/genload.log \$(INTEL_DIALOGIC_DIR)/log/iptconf.log \$(INTEL_DIALOGIC_DIR)/log/oam.log
Demos	%INTEL_DIALOGIC_DIR%\log\rgademo.log %INTEL_DIALOGIC_DIR%\log\Board[#].log	\$(INTEL_DIALOGIC_DIR)/log/rgademo.log \$(INTEL_DIALOGIC_DIR)/log/Board[#].log \$(INTEL_DIALOGIC_DIR)/log/dlgrhdemo.log

Table 1. Log Files Archived by its_sysinfo (Continued)

Log Files	Windows	Linux
RTF log Files	<Logfile path>/rtflog*.txt <Logfile path> found in \$(INTEL_DIALOGIC_DIR)\cfg\RtfConfigwin.xml or \$(DLCFGPATH)\rtfconfig.xml	<Logfile path>/rtflog*.txt <Logfile path> specified in \$(INTEL_DIALOGIC_DIR)\cfg\RtfConfigLinux.xml
	<Logfile path>/rtflog.txt <Logfile path> specified in \$(DLCFGPATH)\rtfconfig.xml	<Logfile path>/rtflog*.txt <Logfile path> specified in \$(DLCFGPATH)/RtfConfigLinux.xml
CASTrace log Files	\$(INTEL_DIALOGIC_DIR)\log\CASttrace.log	\$(INTEL_DIALOGIC_DIR)/log/CASttrace.log.*
Debug Angel Files	\$(INTEL_DIALOGIC_DIR)\bin\DebugAngel.log	\$(INTEL_DIALOGIC_DIR)/log/debugangel.*
Pstndiag Trace log Files	\$(INTEL_DIALOGIC_DIR)\log\pstndiag.*	\$(INTEL_DIALOGIC_DIR)/log/pstndiag.*
IP Protocol Files	\$(SystemRoot)\System32\gc_h3r.log Or \$(INTEL_DIALOGIC_DIR)\bin\gc_h3r.log \$(SystemRoot)\System32\rtvsp1.log Or \$(INTEL_DIALOGIC_DIR)\bin\rtvsp1.log \$(SystemRoot)\System32\sdplog.txt Or \$(INTEL_DIALOGIC_DIR)\bin\sdplog.txt \$(SystemRoot)\System32\siplog.txt Or \$(INTEL_DIALOGIC_DIR)\bin\siplog.txt Note: This is obsolete. IP protocols supports RTF.	\$(HOME)/g*.log \$(HOME)/rtvsp1.log Note: This is obsolete. IP protocols supports RTF.
Dr. Watson Dump and Log Files	\$(USERPROFILE)\Local Settings\Application Data\Microsoft\Dr Watson\DrWtsn32.log	
Its_sysinfo logfile files...	its_sysinfo.log	its_sysinfo.log

- Notes:** 1. Windows: INTEL_DIALOGIC_DIR=C:\Program Files\Dialogic and SystemRoot=C:\WINNT
2. Linux: INTEL_DIALOGIC_DIR=/usr/dialogic/ and HOME=/root

20.5 System Information Data Structuring

Data collected by its_sysinfo will be available as a single compressed file that contains the following:

- A master data file in HTML format.



- 0 or more files as attachments. The HTML file will contain links and explanations of the attachments.
- An application operation log file generated the during data collecting process.
- The operation log files and attachments will be put in the compressed zip file.



This chapter provides reference information about the kernelver tool. The following topics are provided:

- [Description](#). 91
- [Options](#). 91

21.1 Description

The KernelVer tool queries the board's kernel running on a particular processor for its version number. This tool can be used to verify whether or not a processor has crashed.

21.2 Options

The kernelver tool uses the following command line options:

- b<n>
Logical ID of board (required). Use the listboards utility (Linux) or the Configuration Manager (DCM) (Windows) to obtain the board's logical ID.
Note: The listboards utility is described in the *Administration Guide* for the release and the Configuration Manager is described in the *Configuration Guides* for the (Windows) release.
- d<level>
Do not modify. Leave this at the default value of 0.
- f<file>
Output file name (required to save output in a file).
- p<n>
Processor number (required). Lowest allowable value is 1.
- l<n>
Number of times the program will retrieve the version (optional). You can use this option to repeatedly ping the board's kernel to generate message traffic.
- h
Displays the help screen.
- v
Displays the version number.

The following example runs the KernelVer tool on board 1, processor 2:

```
kernelver -b1 -p2
```


This chapter provides reference information about the PDK Trace tool. The following topics are included:

- Description. 93
- Guidelines 93
- Options. 94
- Sample Scenarios. 94

22.1 Description

The DM3 PDK Protocol Trace (PDK Trace) tool allows those who use a DM3 PDK protocol to log specific information related to the operation of the protocol. The PDK Trace tool is useful for protocol developers because it can trace the runtime states, input signals, output signals, and decision branches of the SDL protocol. The PDK Trace tool allows you to specify which channels on the board to begin tracing. This can be a single channel on one trunk, a single channel on multiple trunks, a range of channels on one trunk, or a range of channels on multiple trunks.

PDK Tracing will work on any of the following products with a PDK protocol loaded: Intel NetStructure® DM/V, DM/V-A, DM/V-B, DM/F, and DMT160TEC boards. PDK tracing is not supported for Intel NetStructure® on DM3 architecture analog boards and Intel NetStructure® DM/IP boards.

Note: If you are using an Intel NetStructure DM/IP board and get the following error message, you can ignore it (the error message appears in DebugAngel - see [Chapter 11, “DebugAngel Reference”](#)):

```
001:CP1:Cause-Tag : 80013, Error File :qkernerr.h.-Failed to find tracer component address
```

22.2 Guidelines

The PDK Trace tool requires the Global Call Protocols, so you must choose the Global Call Protocols option when you install the Intel® Dialogic® System Release software. Refer to the *Installation Guide* for details.

The PDK Trace tool's executable name is *pdktrace.exe* for the Windows operating system and *pdktrace* for the Linux operating system. The PDK Trace tool is a command line application that will create a system process thread to interact with the DM3 system to capture trace data. This data will be streamed to a file on the host system.

Upon completion of tracing, the data will be stored on the host system in a file specified when tracing was started (default: *pdktrace.log*). The file is in an unreadable binary format and will need to be converted to a readable format. For help with this, contact technical support (<http://resource.intel.com/telecom/support/contact.htm>).

22.3 Options

The PDK Trace tool uses the following command line options:

-b#

This **required** option specifies the logical board ID of the board to trace.

Example: `pdktrace -b0` where 0 is the logical board ID of the destination board.

-l[#] or -l[#-#]

Specify which trunk(s) the channels to be traced are located on. The default value is 1 (trunk 1).

Example 1: `pdktrace -b0 -l[1] /* Single trunk */`

Example 2: `pdktrace -b0 -l[1-4] /* Range of trunks */`

-c[#] or -c[#-#]

Specify which channel(s) on the specified trunks to trace. The default value is 1 (channel 1).

Example 1: `pdktrace -b0 -l[1] -c[5] /* Single channel */`

Example 2: `pdktrace -b0 -l[1-4] -c[1-30] /* Range of channels */`

-f[filename]

Specify the name of a file on the host system to write the trace data to. The default is *pdktrace.log*.

-i

This option is **required** only when you **first** use the PDK Trace tool. This option is used to initialize the DM3 Tracer Component in the firmware.

Note: This option should only be used the first time the utility is executed after the board is downloaded

-v

Prints the version number of the utility.

-, -h

Prints the help screen (command line options) for the utility.

Warning: Due to memory constraints in the embedded DM3 system, the tool will limit the number of channels that can be traced simultaneously to 60. Trying to trace more than 60 channels per board can cause unpredictable results.

22.4 Sample Scenarios

The following are some sample scenarios in which the PDK Trace tool can be used and the command line options used to specify each configuration. These sample scenarios assume that the logical ID of the board being used is 0.

Scenario 1: For board 0, trace channel 1 on trunk 1

```
pdktrace -b0
pdktrace -b0 -l[1]
pdktrace -b0 -l[1] -c[1]
```

Scenario 2: For board 0, trace channels 10-20 on trunk 1

```
pdktrace -b0 -c[10-20]  
pdktrace -b0 -l[1] -c[10-20]
```

Scenario 3: For board 0, trace channel 1 on trunks 1-4

```
pdktrace -b0 -l[1-4]  
pdktrace -b0 -l[1-4] -c[1]
```

Scenario 4: For board 0, trace channels 1-24 on trunks 1-4

```
pdktrace -b0 -l[1-4] -c[1-24]
```

Note: If any of the above scenarios were being executed on a board for the first time after it was downloaded, the `-i` command line option should also be used as follows:

```
pdktrace -b0 -l[1-4] -c[1-24] -i
```



This chapter provides reference information about the Phone tool. The following topics are included:

- [Description](#) 97
- [Guidelines](#) 97
- [Options](#) 97

23.1 Description

The Phone tool uses the TSC and ToneGen instances and requires a TSC component. The Phone tool can control a single DM3 resource channel (make calls, wait for calls etc.), monitor channel and call states, and send call control operations to a DM3 Global Call resource.

23.2 Guidelines

Phone is a QScript utility. For more information about QScript utilities, refer to [Chapter 25](#), “QScript Reference”.

When using Dynamic Routing configurations, the audio buttons in tools such as the Phone tool will not work.

23.3 Options

The Phone tool uses the following command line options:

-b<n>

Logical ID of board (required). Use the listboards utility (Linux) or the Configuration Manager (DCM) (Windows) to obtain the board's logical ID.

Note: The listboards utility is described in the *Administration Guide* for the release and the Configuration Manager is described in the *Configuration Guides* for the (Windows) release.

-l<n>

Line number (optional). The default value is 1.

-chan<n>

Channel number (optional). The default value is 1.

The following example runs the Phone tool on board 1, line 1:

```
phone -b1 -l1
```



PSTN Diagnostics Tool Reference **24**

This chapter provides the following reference information about the PSTN Diagnostics tool (pstndiag):

- [Description](#) 99
- [Guidelines](#) 100
- [Preparing to Run the PSTN Diagnostics Tool](#) 100
- [Running the PSTN Diagnostics Tool](#) 100
- [Checking the pstndiag Log File](#) 107
- [PSTN Diagnostics Menus](#) 108
- [PSTN Diagnostics Command Line Options](#) 109

For information about how to use the pstndiag tool to perform diagnostic tasks, refer to these chapters:

- [Chapter 5, “Diagnosing First Call Issues”](#)
- [Chapter 6, “Diagnosing PSTN Protocol Issues”](#)

24.1 Description

The PSTN Diagnostics tool (pstndiag) is a utility for diagnosing and troubleshooting public switched telephone network (PSTN) connectivity problems on specific hardware products based on DM3 architecture.

The pstndiag tool allows you to perform the following activities:

- Query the system for board information
- View all boards in the system that have been downloaded
- Monitor all components (boards, trunks, channels) in the system, and view trunk and channel information, such as trunk status, alarm status, channel state, and call state
- Display the protocol family running on a channel (ISDN or CAS)
- Produce a consolidated log file for all components that are being actively monitored in the system
- Display a previously saved log file
- Launch the ISDNtrace and CASrtrace tools to perform further diagnostic tests

24.2 Guidelines

The following restrictions apply to the pstndiag tool:

- The tool is not supported on boards based on Springware architecture.
- It is recommended that you view or monitor no more than 8 channels in your system at one time. Viewing more than 8 channels at one time may negatively impact system performance.
- The pstndiag tool will only identify and list boards with PSTN front-end interfaces. Boards that do not have PSTN front-end interfaces (for example, resource only) will not be presented and cannot be monitored or manipulated by pstndiag.

Pstndiag is a QScript utility. For more information about QScript utilities, refer to [Chapter 25, “QScript Reference”](#).

24.3 Preparing to Run the PSTN Diagnostics Tool

Before running the pstndiag tool, ensure that you have performed the following tasks:

1. The Intel® Dialogic® system release software has been properly installed.
2. The Intel Dialogic system has been started for the boards in your system. The pstndiag tool will detect boards that have already been downloaded.

24.4 Running the PSTN Diagnostics Tool

This section provides instructions for running the pstndiag tool and describes the user interface. The following topics are covered:

- [Starting the PSTN Diagnostics Tool](#)
- [The Main Window](#)
- [The View Bar](#)
- [Option Buttons and Command Buttons](#)
- [Keyboard Navigation](#)

24.4.1 Starting the PSTN Diagnostics Tool

This section contains instructions for starting the pstndiag tool in Windows and in Linux.

To start the pstndiag tool in **Windows**, follow these instructions:

1. Open a command prompt window. The default location of the tool is `%INTEL_DIALOGIC_DIR%\qscript\tools\pstndiag\`. However, you can run the tool from any location.
2. At the command prompt, type:
`pstndiag`

To start the pstndiag tool in **Linux**, type `pstndiag` from the command line of a `cmd` prompt. No arguments are required because the path is set by the install of the Intel Dialogic System Release software.

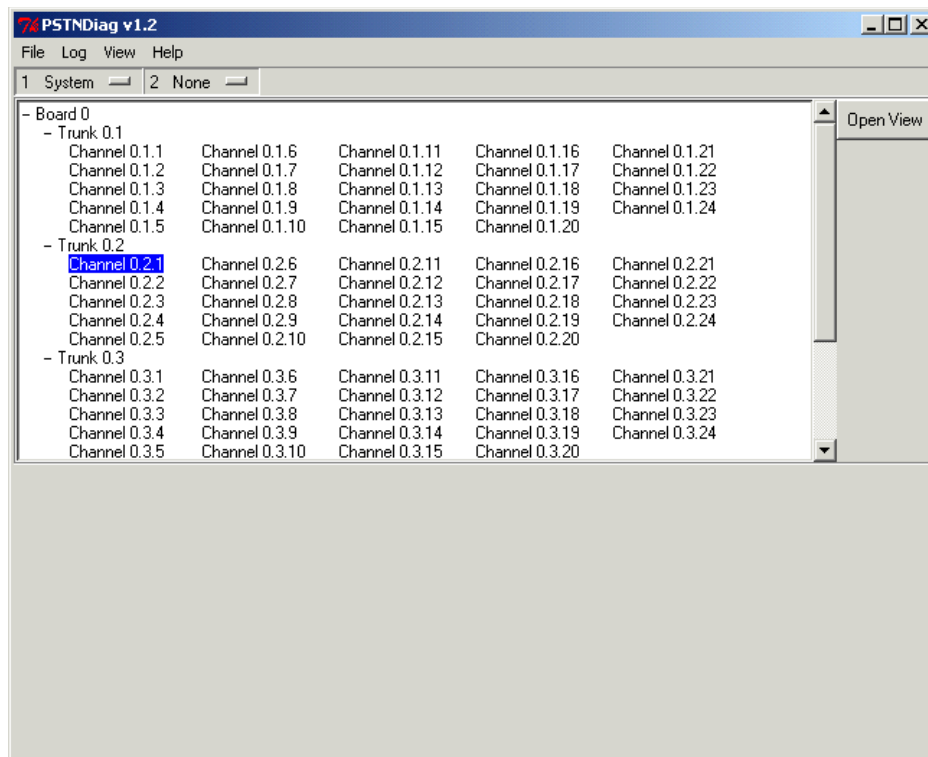
- Notes:**
1. You can specify options at the command line. For more information on these options, see [Section 24.7, “PSTN Diagnostics Command Line Options”](#), on page 109.
 2. If a board is stopped after you have launched the pstndiag tool, you will need to exit from the tool, restart the board, and relaunch the tool to use it again.

24.4.2 The Main Window

After you have launched the pstndiag tool, the **main window** displays a hierarchical tree of all known components in your system. This view is called the **system level view**. The tree consists of three levels: board, trunk, and channel. At the top level, the tree lists all boards that have been downloaded in your system. For each board, all trunks (or spans) are listed. For each trunk, all channels within a trunk are listed. Each of these components (board, trunk, channel) is called a device. The tree can be collapsed or expanded by clicking on the + and - icons.

Figure 3 illustrates a system level view of a DM/V960A-4T1-PCI board.

Figure 3. PSTN Diagnostics Tool - System Level View



The numbering convention, x.y.z, is used to identify a component in the tree, for example “**Channel 0.2.1**”, where:

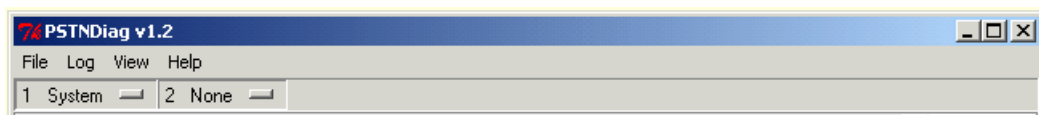
- x is an integer that represents the **board** logical ID, in this example **0**
- y is an integer that represents the **trunk** on that board, in this example **2**
- z is an integer that represents the **channel** number on that trunk, in this example **1**

If only one number is used in the identifier, this number represents the board, such as Board 0 and Board 1. If two numbers are used in the identifier, such as Trunk 0.2 and Trunk 1.5, the first number represents the board and the second number represents the trunk.

24.4.3 The View Bar

The **view bar** contains two option buttons: one (1) represents the upper pane, and the other (2) represents the lower pane. When you first launch the pstndiag tool, the system level view is displayed in the upper pane; the lower pane is empty. The view bar shows these two option buttons: “**1 System**” and “**2 None**”. Figure 4 illustrates a sample view bar.

Figure 4. PSTN Diagnostics Tool - View Bar



The label on the option button changes according to what is displayed in that pane. Once a component has been opened, it is then available from both option buttons. To display a component in the upper pane or lower pane, select it from the appropriate option button. To close a view, right-click on the component in the option button and click close view. Alternatively, double-clicking on a component in the system level view displays that component in the upper pane. Shift-double-clicking displays the component in the lower pane.

A view can display a system, a board, a trunk, a channel, or a log file. For example, in a back-to-back test scenario, you might want to display two channel level views at one time. Or you might want to display a board level view and a log file view.

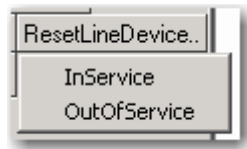
24.4.4 Option Buttons and Command Buttons

Option buttons are buttons that provide additional options. These buttons are identified by a short horizontal bar, as seen in Figure 7 through Figure 14. Click on a button to view additional options, and click on an option to select it. An action occurs, or a new set of parameters is displayed. The use of option buttons allows related information to be grouped together and helps to reduce clutter on the screen.

Command buttons are buttons that when pressed activate a command. There are two types of command buttons: those that toggle between two actions (these are identified by two periods following the button name) and those that issue a command directly. For example, the

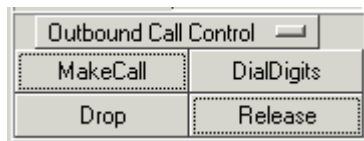
ResetLineDevice.. button in Figure 5 toggles between setting a channel in service and out of service.

Figure 5. Toggle Command Buttons



Examples of command buttons that issue a command directly are the MakeCall, DialDigits, Drop, and Release buttons in a channel level view shown in Figure 6. Click on a button to issue a command.

Figure 6. Direct Command Buttons



Following is a list of the option buttons. There is a submenu of command buttons under each of the option buttons.

Channel State Commands

The commands under this button are as follows:

SetChanState

Set a channel in service or out of service

ResetLineDevice

Drop all calls on the channel and return the call state to Idle

Figure 7. Channel State Command Buttons



Outbound Call Control Commands

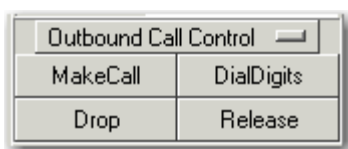
The commands under this button are as follows:

MakeCall

Place an outgoing call

- Drop
Delete a call
- DialDigits
Generate signals for selecting and establishing a connection
- Release
Release the active call on the channel.

Figure 8. Outbound Call Control Command Buttons

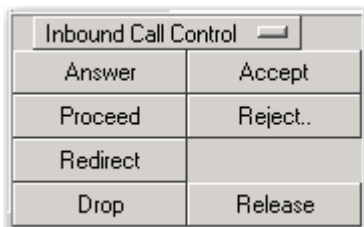


Inbound Call Control Commands

The commands under this button are as follows:

- AnswerCall
Answer the call
- AcceptCall
Accept the call (this precedes the option of handing off the call to another client or party)
- ProceedCall
Proceed the call (this sends the call proceeding message to the originating side, and waits for Accept (to accept call) or Answer (to answer call) as a subsequent command from the host for further progress of the call). This is available only for ISDN.
- RejectCall
Reject the call
- RedirectCall
Redirect the call elsewhere.

Figure 9. Inbound Call Command Buttons



Call Hold Commands

Note: The following commands are advanced functions for technical support use only.

The commands under this button are as follows:

HoldCall

Place a call on hold. A call that is on hold is disconnected from the external receive and transmit data streams that are associated with the Telephony Service Channel (TSC) instance. Once the call is on hold, the client is free to establish a new call, either by making an outbound call, or by receiving an inbound call.

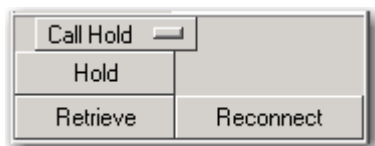
RetrieveCall

Retrieve a call that is on hold, and bring it to the active state. The completion of the Retrieve Call operation is indicated via a Call State event that reports the call state transition to Connected. Note that if there is a call active when the RetrieveCall command is issued, then that call will be placed on hold, so the command can be used to alternate between two calls.

ReconnectCall

An alternative to the RetrieveCall command, ReconnectCall will retrieve a call that is on hold and make it the active call. The difference is that if a call is active when the command is issued, that call will be dropped rather than put on hold.

Figure 10. Call Hold Command Buttons



Call Transfer Commands

Note: The following commands are advanced functions for technical support use only.

The commands under this button are as follows:

BlindTransfer (single-stage blind-transfer)

Place the target call on hold, make a consultation call to the destination address, and then hand off the target call to the destination without determining the outcome of the consultation call. The target call's call state will transition first to HoldPendXfer, and then to Idle.

InitTransfer (part of multi-stage transfer)

Place the target call on hold (HoldPendXfer) and initiate a consultation call to the destination address.

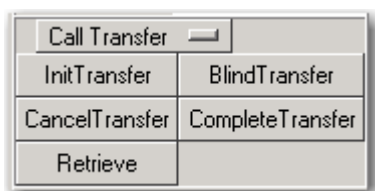
CompleteTransfer (part of multi-stage transfer)

Transfer the call that was the target of the InitTransfer command to the destination address.

CancelTransfer (part of multi-stage transfer)

Cancel a supervised transfer by dropping the consultation call and reconnecting to the call that was the target of the transfer operation.

Figure 11. Call Transfer Command Buttons



SendISDN Command

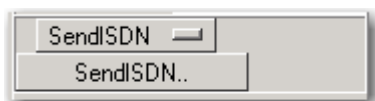
Note: The following command is an advanced function for technical support use only.

The command under this button is as follows:

SendISDN

Sends an arbitrary call-associated Q.931 ISDN Message.

Figure 12. SendISDN Command Button



Misc. Commands

Note: The following commands are advanced functions for technical support use only.

The commands under this button are as follows:

Complete

This command is not currently used.

CancelComplete

This command is not currently used.

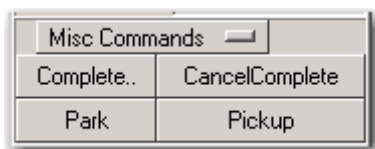
Park

This command is not currently used.

Pickup

This command is not currently used.

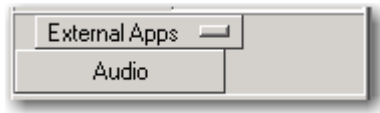
Figure 13. Misc. Commands Buttons



External Applications

The Audio command is not functional.

Figure 14. External Application Command Button



24.4.5 Keyboard Navigation

You can use the following keyboard keys to navigate through the pstndiag tool:

Up Arrow/Down Arrow

Scrolls through devices in the tree.

Left Arrow/Right Arrow

Collapses or expands a level in the tree.

Return or Double-click

When a component is highlighted, creates a view for this component in the upper pane.

Shift-Return or Shift-Double-click

When a component is highlighted, creates a view for this component in the lower pane.

Right-click

When a button in the View bar is highlighted, displays the close view option.

24.5 Checking the pstndiag Log File

This section describes the log file generated by the pstndiag tool.

Logging starts automatically when you launch the pstndiag tool. Logging information is written to the log file for any component (board, trunk, and channel) that is being monitored and opened in a view. The log file is in binary format and must be viewed in the pstndiag tool.

Caution: The diagnostics log file has no maximum size. After completing diagnostics tasks, you should exit the pstndiag tool. Do not run the pstndiag tool indefinitely; otherwise, the resulting log file may consume a large amount of disk space.

The default log file name is *pstndiag.yyyymmdd.hhmmss.pdlog*, where *yyyy* is the year, *mm* is the month, *dd* is the day, *hh* is the hour, *mm* is the minute, and *ss* is the second when the file is created. Each time the tool is run, a new log file is created. Running multiple copies of the tool will result in multiple log files. The log file is written by default to the standard log directory:

- Windows - %INTEL_DIALOGIC_DIR%/log
- Linux - \${INTEL_DIALOGIC_DIR}/log

The log file generated by the pstndiag tool captures all events associated with your use of this tool. There are two types of events: application events (AppEvent) and message events (MsgEvent). The application events are generated as actions occur in the pstndiag tool; for example, line out of service. Message events are messages from the firmware in response to application events or asynchronous events; for example, trunk active. If you run other applications such as demos, only message events are generated. The log file also captures application errors (AppErrors).

A time stamp is provided for each event as it occurs. The event is associated with a board, trunk, or channel. For example, AppEvent(0.2.1) is associated with board 0, trunk 2, and channel 1. The log file also provides a graphical view of each event as it occurs over time.

To display the current log file, select View running log from the Log menu. To view a previously saved log file, select Open log file from the File menu.

24.6 PSTN Diagnostics Menus

This section provides a reference to the menus available in the PSTN Diagnostics or pstndiag tool:

- [File menu](#)
- [Log menu](#)
- [View menu](#)
- [Help menu](#)

24.6.1 File menu

The File menu contains the following menu options:

Open log file

Opens a previously saved log file.

Exit

Exits the tool.

24.6.2 Log menu

The Log menu contains the following menu options:

View running log

Displays the current log file in a window.

Logging active

Enables logging. The default setting is logging enabled. To disable logging, you can use the “-nolog” option at the command line. For information on command line options, see [Section 24.7, “PSTN Diagnostics Command Line Options”](#), on page 109.

24.6.3 View menu

The View menu contains the following menu options:

Open system view

Opens a system level view.

Open new view

Opens a board, trunk, or channel level view.

24.6.4 Help menu

The Help menu contains the following menu options:

About

Displays the version number of the pstndiag tool as well as copyright information.

Command line

Displays information about command line options.

General information

Displays information about menus, menu options, and views.

24.7 PSTN Diagnostics Command Line Options

Command line options enable you to bypass the system level view and display a specific board, trunk, or channel level view when you start the pstndiag tool.

The following command line options are available:

-b

Starts with a specified board level view, where represents the board logical ID.

Example: -b3 will display the board level view for Board 3 in your system.

-board

Same as -b.

-c.<t>.<c>

Starts with a specified channel level view, where represents the board logical ID, where <t> represents the trunk (span) on the board, and where <c> represents the channel.

Example: -c3.1.24 will display the channel level view for channel 24 on trunk 1 of board 3.

-chan .<t>.<c>

Same as -c.<t>.<c>.

-t.<t>

Starts with a specified trunk level view, where represents the board logical ID and where <t> represents the trunk on that board.

Example: -t3.1 will display the trunk view for trunk 1 of board 3.

-trunk .<t>

Same as -t.<t>.

- help
Displays help information for the tool.
- logto <file>
Saves log information from open views to the specified file name.
- nolog
Does not automatically log events for open views.
- noscan
Does not perform an initial system scan to determine available resources. All views must be opened manually in this mode. Using this option can reduce the time it takes for the tool to start.
- openlog <file>
Starts by displaying the specified log file. The log file extension is *.pdlog* (this file must be viewed in the pstndiag tool). When specifying a directory path, use forward slash.
Example: -openlog c:/temp/test.pdlog

This chapter provides information about QScript utilities, which are a subset of the diagnostic utilities.

- Description. 111
- Guidelines 111
- To use a QScript utility, use the script file. Specify the utility name and parameters on the command line. The location of the script files used to invoke the QScript tools is: . . . 112
- QScript Environment Variables 112

25.1 Description

The QScript utilities are a subset of the diagnostic utilities. QScript is an object-oriented scripting tool developed for the Intel NetStructure® on DM3 architecture boards.

The following diagnostic utilities use QScript:

- CallInfo
- DigitDetector
- Phone
- PSTN Diagnostics or pstndiag (uses several QScript utilities)

The following administrative utilities, which are documented in the System Release *Administration Guide*, use QScript:

- Line Administration Utility (lineadmin)
- Standard DM3 Configuration Utility (stdconfig)
- TSP Configuration Utility (tspconfig)
- TSP Monitor Utility (tspmon)
- TSP Tracer Utility (tsptrace)

25.2 Guidelines

This section describes guidelines for using QScript utilities.

QScript utilities use board and line numbers as follows: board numbers are the logical board ID (1-based), which can be obtained with the listboards utility. Line numbers are also 1-based. That is, the first board is typically board 1 and the first line is line 1.

Note: The listboards utility is described in the *Administration Guide* for the release.

To use a QScript utility, use the script file. Specify the utility name and parameters on the command line. The location of the script files used to invoke the QScript tools is:

```
${INTEL_DIALOGIC_DIR}/bin
```

The QScript tools developed by Intel are located in:

```
${INTEL_DIALOGIC_QSCRIPT}/tools
```

Note: Do not run a *<toolname>.qs* file directly from this directory. Use the script file located in the script file directory, which calls the QScript interpreter to run the *<toolname>.qs* file.

The QScript tools developed by Intel use classes in:

```
${INTEL_DIALOGIC_QSCRIPT}
```

25.3 QScript Environment Variables

This section describes the QScript environment variables:

- [INTEL_DIALOGIC_QSCRIPT Environment Variable](#)
- [Single Session Variable](#)

INTEL_DIALOGIC_QSCRIPT Environment Variable

To run the QScript tools in a Linux environment, the QSCRIPT_DIR environment variable needs to be set to:

```
${INTEL_DIALOGIC_QSCRIPT}
```

Single Session Variable

Set the variable for a single session using the `setenv` command (C Shell) or `set` and `export` command (K and Bourne Shell). To permanently set the environment variable for all login sessions, update the user's profile file to include the variable and associated values.

This chapter describes how to use the Runtime Trace Facility Manager GUI (RTFManager) to configure the trace levels and output of the Runtime Trace Facility (RTF) utility. This chapter contains the following information:

- [Description](#) 113
- [Guidelines](#) 114
- [Starting RTFManager](#) 114
- [Main Window](#) 114
- [General Tab](#) 116
- [Filtering Tab](#) 118
- [Log File Filtering](#) 120
- [Advanced Tab](#) 121

If you prefer to configure RTF manually or need a detailed explanation of what the RTF configuration file contains, refer to [Chapter 27, “Runtime Trace Facility \(RTF\) Reference”](#).

26.1 Description

RTF provides a mechanism for tracing the execution path of the runtime libraries for the Intel® Dialogic® system release software. All libraries that use RTF write their trace messages to a log file. The resulting log file helps troubleshoot runtime issues for applications that are built with Intel Dialogic system release software.

Trace control settings and log file formatting are set via the Runtime Trace Facility Manager. Each runtime library has several levels of tracing that can be dynamically enabled/disabled through the Runtime Trace Facility Manager.

The Runtime Trace Facility Manager allows you to do the following:

- trace the host libraries on DM3, Springware, and Host Media Processing technology
- use a graphical user interface (Runtime Trace Facility Manager) to configure trace control settings and the format of resulting log files
- use a command line utility (rtftool) to start/stop RTF tracing and incorporate RTF tracing configuration changes dynamically
- trace multiple processes and format trace output into a central data store (log file)
- dynamically configure trace options at runtime without terminating running processes
- adjust the runtime configuration, allowing you to trace select modules (libraries), clients (trunks or channels) and labels (trace categories or levels) to avoid collecting excess trace data and degrading system performance

- filter trace data, allowing you to customize the trace output so that only certain trace statements are stored
- customize the format, readability, and location of the log files
- limit the size of the log files
- define the number of rolling backup log files to keep.

26.2 Guidelines

Guidelines applicable to RTFManager can also be found in [Section 27.4, “Restrictions and Limitations”](#), on page 136. Note that some items in this section are only relevant if you are manually editing the RTF configuration file.

In addition, here are some guidelines specific to RTFManager:

- RTFManager will remember your previous settings when opening the configuration file.
- RTFManager can be used to reload a new trace configuration into RTF while applications are running, thus allowing you to change tracing on the fly. RTF will reload the configuration XML file after you change the configuration either on the [General Tab](#) or [Filtering Tab](#) and click the **OK** button.
- RTFManager enables you to reset the configuration file to its default state (all product families and all technologies tracing at the Error level) by using the **Reset Default Settings** button on the [Filtering Tab](#).
- Sometimes “custom” configuration files are used. RTFManager will detect whether or not the configuration file is custom. Custom configuration files are used for specific troubleshooting purposes and are not usually modified by most users. They can only be modified using the Advanced tab. For more information, refer to [Section 26.8, “Advanced Tab”](#), on page 121.

26.3 Starting RTFManager

On Windows systems, RTFManager can be started from the Start menu. RTFManager is in the program group for the Intel telecom software (System Release or Host Media Processing). RTFManager can also be started by just typing “RTFManager” from the console since it is in the path of the operating system’s executable searching.

On Linux systems, RTFManager can be started by typing `/usr/dialogic/RTFManager` or `RTFManager` if the executable is in the operating system’s executable searching path.

26.4 Main Window

After you have launched the Runtime Trace Facility Manager, the main window appears. The main window allows you to view existing log files and set Runtime Trace Facility configuration options. The main window toolbar contains the following menu items:

- [File](#)

- [Edit](#)
- [Tools](#)
- [Help](#)

26.4.1 File

The File menu contains options for working with the log files. The File menu contains the following options:

Open

Opens a previously saved log file.

Close

Closes the open log file.

Save

Saves the open log file.

Save As

Saves a copy of the log file under a different name or in a different location.

Exit

Closes the Runtime Trace Facility Manager.

26.4.2 Edit

The Edit menu contains options for manipulating and searching the text of an existing log file. The Edit menu contains the following options:

Cut

Cuts the selected text.

Copy

Copies the selected text.

Paste

Pastes text to the selected location.

Delete

Deletes the selected text.

Find...

Launches the Find dialog box. The Find dialog box allows you to search the log file for alphanumeric strings.

Find Next...

Finds the next instance of an alphanumeric string.

Replace...

Replaces the selected text with new text.

Go To...

Jumps to a given line number in the log file.

Select All

Selects all text in the log file.

26.4.3 Tools

The Tools menu contains options for filtering the log file that is displayed in the Runtime Trace Facility Manager main window and configuring the RTF trace capabilities. The Tools menu contains the following options:

Filter...

Launches the Filters dialog box. The Filters dialog box allows you to selectively display certain trace statements in the log file. Refer to [Section 26.7, “Log File Filtering”](#), on page 120 for more information.

RTF Configuration...

Launches the Runtime Trace Facility Manager Configuration window. The Runtime Trace Facility Manager Configuration window contains the following two tabs:

- General - The General tab is used to enable/disable RTF tracing and customize the format, readability and location of the log files. Refer to [Section 26.5, “General Tab”](#), on page 116 for more information.
- Filtering - The Filtering tab allows you to determine which libraries to trace and the trace level (Debugging, Program Flow, Warnings, Errors) for each library. Refer to [Section 26.6, “Filtering Tab”](#), on page 118 for information about using the Filtering tab.

26.4.4 Help

The Help menu contains the following options:

Help Topics

Launches the Runtime Trace Facility Manager online help.

About

Displays the version number of the Runtime Trace Facility Manager.

26.5 General Tab

The General tab is used to enable/disable RTF tracing and customize the format, readability and location of the log files.

Note: It is possible to reload a new trace configuration into RTF while applications are running. Just change the configuration on RTFManager’s General Tab and click the **OK** button. RTF will then reload the configuration XML file.

The general tab contains the following settings:

Enable Global Tracing?

This check box is used to enable or disable the RTF tracing capabilities. If the box is checked, RTF tracing is enabled. If the box is not checked, RTF tracing is disabled.

Note: Enable/disable means RTFManager will modify the RTF configuration file to change certain values to enable/disable a group of modules. RTFManager will reload the RTF to let the change take effect.

Timestamp

This check box acts as a Boolean flag to determine whether or not time stamps are generated in the log file. If the box is checked, time stamps will appear in the log files. If the box is not checked, time stamps will not appear in the log files.

Log Format

This attribute defines the format of the log file. The following three values are supported:

- Text Aligned - Aligns the trace entry fields in the log file into comma separated columns. The top of each column includes a header that provides a description of the column's content. This is the default setting.
- Text Unaligned - Separates the trace entry fields in the log files with commas only. Column alignment is not done.
- Binary - Not supported by the current release.

Note: Using the Text Aligned setting makes the log file easier to read but it does not make efficient use of hard drive space. This inefficiency is exacerbated as the log file grows. The Text Unaligned format is separated by commas so it can be parsed by a spreadsheet or database program to make the file easier to read.

Logfile Directory

This attribute defines the location of the log file. This attribute is read-only.

Logfile Prefix

This attribute defines the name of the log file. This attribute is read-only.

Logfile Size

Sets the maximum size, in Kilobytes (KB), of the log file. The default setting is 1000. The size is the maximum size of a single log file. When the file reaches this size, the RTF behavior depends on the setting of the attribute as follows:

- If Maximum Backups is set to 0, RTF “rolls over” the log file, much like a circular buffer. RTF aligns the entries so that the oldest entry is always at the top of the log file and the newest entry is always at the bottom of the log file.
- If Maximum Backups is set to 1 or greater, the log file will be closed and a new log file will be created. This sequence continues until the Maximum Backups threshold is reached.

Maximum Backups

This attribute is used to determine whether or not backup log files are created. Valid values are as follows:

- 0 - Backup log files will not be created. All trace information will be written to one log file. When the size of this log file reaches the threshold defined in the attribute, RTF “rolls over” the log file, much like a circular buffer. RTF aligns the entries so that the oldest entry is always at the top of the log file and the newest entry is always at the bottom of the log file. This is the default setting.

- 1 or greater - Backup log files will be created. All trace information is initially written to one log file. When the size of this log file reaches the threshold defined in the Logfile Size attribute, the initial log file is saved and the trace data is written to a second log file. This sequence occurs until the threshold is reached.

Note: There is no limit to the maximum backups imposed by the RTF itself. However, a higher limit will increase the usage of disk space and impact performance since RTF needs to keep track of all backup files and remove old ones as necessary.

Module Width

Allows you to customize the number of characters that appear in the log file's Module column. This attribute is only applicable when the attribute is set to Text Aligned. The default setting is 10.

Client Width

Allows you to customize the number of characters that appear in the log file's Client column. This attribute is only applicable when the attribute is set to Text Aligned. The default setting is 10.

Label Width

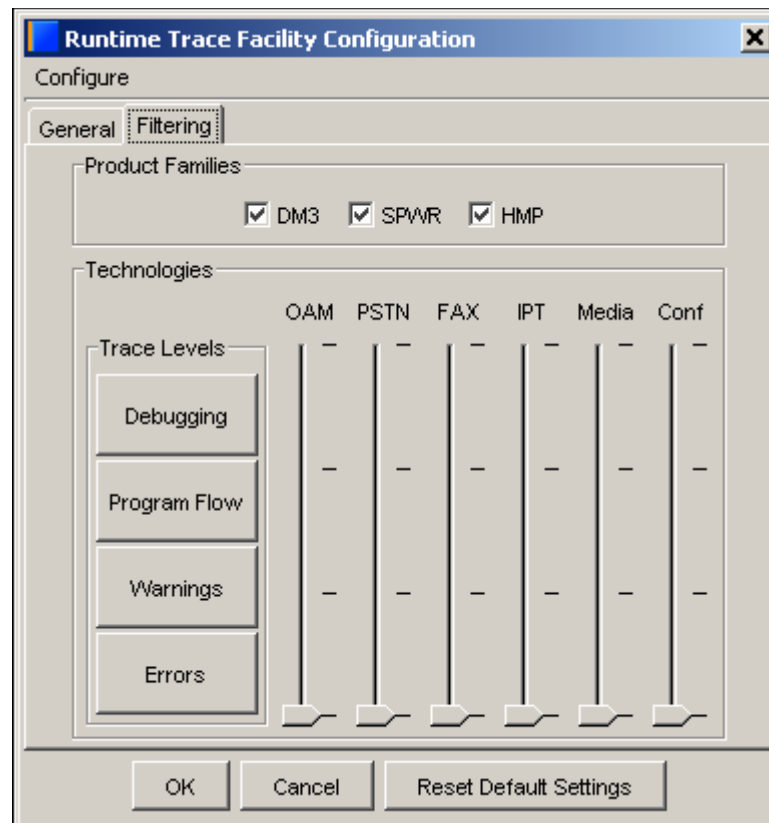
Allows you to customize the number of characters that appear in the log file's Label column. This attribute is only applicable when the attribute is set to Text Aligned. The default setting is 10.

26.6 Filtering Tab

This section provides information about the Runtime Trace Facility Manager Configuration window's Filtering tab, which is shown in Figure 15. The Filtering tab allows you to trace select Intel® Dialogic® System Release libraries and set the trace level for each library. This helps to avoid collecting excess trace data and degrading system performance.

Note: It is possible to reload a new trace configuration into RTF while applications are running. Just change the configuration on RTFManager's Filtering Tab and click the **OK** button. RTF will then reload the configuration XML file.

Figure 15. RTFManager Filtering Tab



The following procedure describes how to use the Runtime Trace Facility Manager Configuration window's Filtering tab:

1. The top row of check boxes represents several Intel® telecom product families (DM3, Springware, Host Media Processing). To trace a library that runs on a given product family, you must populate the check box for the product family. For example, if the application you'd like to trace uses the voice library on Springware hardware, you must check the SPWR box.

Each technology has a slider that allows you to set the RTF's trace level. This is the Intel telecom product technology level. The technology sliders are defined as follows:

- OAM - traces the DM3 transport layer, and also includes such things as timeslot configuration, CT Bus clocking agent, event services (fault detectors), and OAM IPC
- PSTN - traces call control libraries (Global Call, DTI, Standard Runtime)
- FAX - traces Fax libraries
- IPT - traces Internet Protocol libraries (IPML, DM3 transport layer, IP call control)
- Media - traces media libraries (Voice, CSP)
- Conf - traces conferencing libraries (CNF, DM3 transport layer, DCB, MSI, HDSI)

2. Trace levels appear on the left side of the window. Each trace level corresponds to turning on certain labels according to the following:

- Errors - Traces the following labels: ERR1, ERR2, EXCE, and EINF. This is the lowest slider setting, so these tags are always traced.
- Warnings - Traces the WARN label in addition to the Error labels.
- Program Flow - Traces the INTF, INFO, and APPL labels in addition to the Warning and Error labels.
- Debugging - Traces the DEBG label in addition to the Program Flow, Warning, and Error labels. All trace labels are enabled at this level.

Note: Alternatively, you can click on any one of the label buttons to set all sliders to the given level.

3. When you have finished setting the trace levels, click **OK**.
4. You can then start your application (if it not already running). Trace output will appear in a log file. You can display the log file in the Runtime Trace Facility Manager main window using the **File -> Open** menu item. Furthermore, you can use the Runtime Trace Facility Manager main window to only display select trace output statements.

26.7 Log File Filtering

The Runtime Trace Facility Manager provides a log file filtering mechanism that allows you to customize the log file so that only certain trace statements are displayed. Use the following procedure to filter the log file:

1. In the Runtime Trace Facility Manager main window, select **File -> Open** to open an existing RTF log file.
2. Select **Tools -> Filter...** The Filters dialog box appears. The Filters dialog box lists any filters you currently have set up in the Runtime Trace Facility Manager.
3. In the Filters dialog box, click the **New** button. The New Filter window appears.
4. Select **Module**, **Client** or **Label** from the New Filter window's drop-down menu, depending on which element you'd like to view on the log file.
5. In the New Filter window's text box, type the name of the Module, Client or Label you'd like to view in the log file.
6. Click **OK**.
7. The newly created filter will be added to the list in the Filters window. Keep in mind that when more than one filter is shown, they are AND'ed together. For example, if you have the following filters setup:
 - Module is equal to gc
 - Label is equal to DEBG
8. Then the log file will only show the trace statements that contain the gc module and the DEBG label.
9. The log file will refresh and display its contents according to the filters you have set up.
10. If you subsequently modify or delete filters, the original data is re-filtered based on the latest set of filters. The original log file is left unchanged unless you choose to save it using the **File -> Save** or **File -> Save As** menu items.

26.8 Advanced Tab

The Advanced tab is not recommended for most users. Most users should only adjust settings using the [Filtering Tab](#) and [General Tab](#) as described above. Those using the Advanced tab must obtain specific directions on which settings to use. The Advanced tab should primarily be used by Intel developers to modify a custom RTF configuration file. These custom configuration files are used for specific troubleshooting purposes.

RTFManager will detect whether or not the RTF configuration file is custom or not. An RTF configuration file is considered a custom configuration file if it has a label other than the standard labels (ERR1, ERR2, EXCE, EINF, WARN, INTF, APPL, INFO, DEBG) and the state of the label is turned to ON (state="1"). In other words, a custom file is defined as one that does not abide by the rules set up through the checkboxes, technology sliders and level buttons on the [Filtering Tab](#).

If RTFManager detects a custom configuration file, the only way to change the custom configuration file using the RTFManager is to start RTFManager up in advanced mode. Otherwise, the only thing you will be allowed to do is reset the configuration file back to its default and then you will be allowed to use the [Filtering Tab](#) to change the file.

When a custom file is loaded, you are warned that a custom file is being loaded and you must enable the "Advanced Wizard" to configure the tracing levels. The Advanced Wizard is enabled by running RTFManager with the `-expert` command line argument (in this case, you cannot run RTFManager from the regular start menu item but must run it manually from the command line).

A password is required to enter this advanced mode. When you enter advanced mode, the RTFManager will start and provide a third tab named "Advanced." In advanced mode, you are presented with a tree structure of all modules and labels and you can enable or disable any of them.

The Advanced tab has six buttons for use in performing configuration tasks:

Thumbs Up

Enables a particular lib/module/client/label. If the item is already enabled, the Thumbs Up button will be disabled and the Thumbs Down button will be enabled.

Thumbs Down

Disables a particular lib/module/client/label. If the item is already disabled, the Thumbs Down button will be disabled and the Thumbs Up button will be enabled.

Scissors

Deletes an item.

M

Allows you to add a new Module.

C

Allows you to add a new Client.

L

Allows you to add a new Label.

When you have completed work on configuration, click the **OK** button at the bottom of the screen. You can also click **Cancel** to exit the screen without making changes.

Runtime Trace Facility (RTF) Reference

27

This chapter provides an overview of the Runtime Trace Facility (RTF), including information about editing the RTF configuration file (*RtfConfigLinux.xml* for Linux* and *RtfConfigWin.xml* for Windows*) file to set tracing configuration options. Reference information about the RTF command line options is also included. This chapter contains the following subsections:

- Description. 123
- Installing RTF 124
- RTF Configuration File 124
- Restrictions and Limitations 136
- rtftool Command 138
- Example RTF Configuration Files. 139
- Configuring Remote RTF Logging 142

A procedure for using the tool is provided in [Chapter 7, “Debugging Software”](#).

27.1 Description

RTF provides a mechanism for tracing the execution path of the runtime libraries for the Intel® Dialogic® system release software. All libraries that use RTF write their trace messages to a log file. The resulting log file helps troubleshoot runtime issues for applications that are built with Intel Dialogic system release software.

RTF obtains trace control settings and output formatting from an RTF configuration file (*RtfConfigLinux.xml* for Linux and *RtfConfigWin.xml* for Windows). Each runtime library has several levels of tracing that can be dynamically enabled/disabled by editing the RTF configuration file while your application runs. This allows RTF to be configured to meet specific needs.

Major RTF components include:

- Client library (*librtfmt.dll* for Windows, *librtf.so* for Linux): Dynamic loaded library that provides the software interface to make use of RTF functionality.
- Configuration file (*RtfConfigWin.xml* for Windows, *RtfConfigLinux.xml* for Linux): Editable file that allows you to customize the tracing and output capabilities of RTF (for example, which runtime libraries RTF will trace, the trace levels, location and size of backup log files.)
Note: You must have administrative rights to modify the *RtfConfig*.xml* files.
- Server application (*RtfServer.exe* for Windows, *RtfServer* for Linux): Service that communicates to all RTF clients, receives trace data from clients, and dispatches data to disk.

- Utility program (*rtftool.exe* for Windows, *rtftool* for Linux): Command line application that allows you to execute RTF functionality from command line or shell scripts.
- Utility program (*RtfTrace.exe*): Command line application that allows you to execute RTF functionality from command line or shell scripts. This is provided for backward compatibility only. All of this utility's functionality is covered in *rtftool.exe*.

27.2 Installing RTF

RTF files are installed as part of the Intel Dialogic system software installation. The RTF configuration file's default installation directory is determined by the INTEL_DIALOGIC_CFG environment variable. Refer to the *Intel Dialogic System Release Software Installation Guide* for information about Intel Dialogic system software environment variables. Because the RTF configuration file's **trace** attribute is set to 1, RTF tracing is enabled by default. Other than editing the RTF configuration file to customize trace settings for your development environment, there are no special configuration steps for starting RTF.

27.3 RTF Configuration File

To make full use of RTF, you will want to modify the RTF configuration file (*RtfConfigLinux.xml* for Linux, *RtfConfigWin.xml* for Windows). This configuration file allows you to define which trace messages will be included in the RTF output. This subsection provides some guidelines for modifying the RTF configuration file and reference information about the file's XML tags and attributes.

Note: You must have administrative rights to modify the *RtfConfig*.xml* files.

The RTF configuration file uses several terms that should be understood before attempting to edit the file. The following definitions should be kept in mind as you edit the RTF configuration file:

module

a binary file, typically an executable or a shared object library file (.so). An RTF module corresponds to a library or software module that has internal RTF APIs incorporated into its source code.

client

an entity for identifying a device (e.g. "dxxxB1C1"), component (e.g. "WaveFileSource") or a function (e.g. "**dx_play**()") that is to be traced by RTF.

label

an attribute associated with a trace statement (e.g. "Error", "Warning", "Info", "External API entry", "External API exit" etc.). A trace statement's label is used by the trace data output for categorization purposes.

trace entry

individual entries in the trace data output. The trace data output is typically sent to a file or debug stream.

0

when a 0 appears next to a configuration item in the RTF configuration file, it indicates that the configuration item is disabled.

1

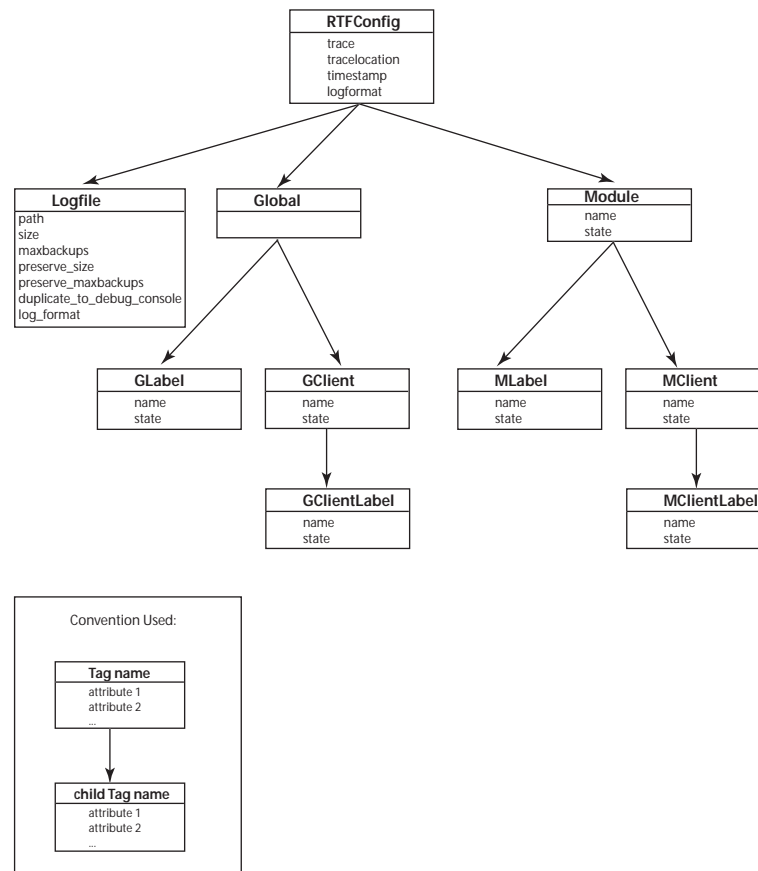
when a 1 appears next to a configuration item in the RTF configuration file, it indicates that the configuration item is enabled.

The RTF configuration file's top-level document tag is the RTFConfig tag. The following three tags are child tags of the RTFConfig tag:

1. Logfile
2. Global
3. Module

Figure 16 shows the XML tag structure and tag attributes of the RTF configuration file:

Figure 16. RTF Configuration File Tag Structure



27.3.1 RTFConfig Tag

RTFConfig is the document tag. This tag is a mandatory component of the RTF configuration file; it can be found at the top of the RTF configuration file. A sample RTFConfig tag entry is shown below:

```
<RTFConfig trace="1" tracelocation="TRACE_LOG" timestamp="1"
logformat="ALIGN">

<!-- Logfile section goes here -->

<!-- Global section goes here -->

<!-- Module sections go here -->

</RTFConfig>
```

The RTFConfig tag includes the following attributes:

trace

This attribute is used to enable or disable the RTF tracing capabilities. Valid values are as follows:

- 0
RTF tracing is disabled.
- 1
RTF tracing is enabled. This is the default setting.

tracelocation

This attribute has no affect on the file and will be removed in a future release. By default, trace output is sent to a log file.

timestamp

This attribute has no affect and will be removed in a future release. By default, each log file name includes a time stamp.

logformat

This attribute defines the format of the log file. The following two values are supported:

ALIGN

Aligns the trace entry fields in the log file into comma separated columns. The top of each column includes a header that provides a description of the column's content. This is the default setting.

UNALIGN

Separates the trace entry fields in the log files with commas. Column alignment is not done.

- Notes:** 1. If you set the logformat attribute to ALIGN, you can customize the widths of the various columns. The following attributes in the RTF configuration file allow you to define the aligned column widths:
- **ModuleWidth number** - Allows you to customize the number of characters that appear in the Module column. The default setting is 10. There is no maximum setting.
 - **ClientWidth number** - Allows you to customize the number of characters that appear in the Client column. The default setting is 15. There is no maximum setting.

- **LabelWidth number** - Allows you to customize the number of characters that appear in the Label column. The default setting is 10. There is no maximum setting.
- 2. Using the ALIGN setting makes the log file easier to read but it does not make efficient use of hard drive space. This inefficiency is exacerbated as the log file grows. The UNALIGN format is separated by commas so it can be parsed by a spreadsheet or database program to make the file easier to read.

27.3.2 Logfile Tag

The Logfile tag is the first child tag of the RTFConfig tag. The Logfile tag provides logistical information about the log file(s) used to store trace output.

If you would like to customize the appearance and settings of the log file, edit the Logfile tag within the RTF configuration file. The Logfile tag appears under the RTFConfig tag in the RTF configuration file.

A sample Logfile tag entry from the *RtfConfigLinux.xml* file is shown below:

```
<Logfile path="$(INTEL_DIALOGIC_DIR)/log" size="300" maxbackups="10"
preserve_size="300" preserve_maxbackups="10"
duplicate_to_debug_console="0" log_format="text"/>
```

A sample Logfile tag entry for the *RtfConfigWin.xml* file is shown below:

```
<Logfile path="$(INTEL_DIALOGIC_DIR)\log" size="300" maxbackups="10"
preserve_size="300" preserve_maxbackups="10"
duplicate_to_debug_console="0" log_format="text"/>
```

The Logfile tag includes the following attributes:

path

Indicates a valid directory path for the log file. The default path for Linux is *\$(INTEL_DIALOGIC_DIR)/log*. The default path for Windows is *\$(INTEL_DIALOGIC_DIR)\log*. The INTEL_DIALOGIC_DIR environment variable is defined as part of the system release software installation routine. Refer to the *Intel Dialogic System Release Software Installation Guide* for more information about the INTEL_DIALOGIC_DIR environment variable.

size

Sets the maximum size, in Kilobytes (KB), of the log file when RTF is run with preservation mode turned OFF. The default setting is 300. When the file reaches its maximum size, RTF “rolls over” the log file, much like a circular buffer. The previous log file is saved as a timestamped text file. RTF aligns the entries so that the oldest entry is always at the top of the log file and the newest entry is always at the bottom of the log file.

- Notes:**
1. Due to the internal buffers used by RTF, the actual size of the log file may be up to 1 MB larger than the size attribute's value. For example, if the size attribute is set to 1000 KB, the actual log file may grow up to 2000 KB.
 2. This attribute is only applicable when RTF preservation mode is turned OFF. When RTF preservation mode is ON, the `preserve_size` attribute determines the size of the log file. Refer to [Section 27.5.3, "Running RTF in Preservation Mode"](#), on page 138 for information about preservation mode.

maxbackups

Indicates the maximum number of backup log files the RTF creates when the RTF is run with preservation mode turned OFF. If this attribute is set to 0, all trace information is written to one log file. If this attribute is set to 1 or greater, all trace information is initially written to one log file. When the size of this initial log file reaches the threshold defined in the Logfile tag's size attribute, the RTF trace data "rolls over" into a second log file. When the size threshold is reached in the second log file, the RTF trace data "rolls over" into a third log file. This sequence occurs until the number of backup log files created equals the maxbackups attribute setting. When the "roll over" occurs, the previous log file is saved under the following timestamped name:

`rtflog-<year><month><day>-<hour>h<minute>m.<second>s.txt`

If the number of backup log files exceeds the number defined in maxbackups, the oldest backup file will be deleted and a new one will be created, on a first in, first out basis. The default maxbackups value is 10.

- Notes:**
1. This attribute is only applicable when RTF preservation mode is turned OFF. When RTF preservation mode is ON, the `preserve_maxbackup` attribute determines the number of backup log files. Refer to [Section 27.5.3, "Running RTF in Preservation Mode"](#), on page 138 for information about preservation mode.
 2. There is no limit to the maximum backups imposed by the RTF itself. However, a higher limit will increase the usage of disk space and impact performance since RTF needs to keep track of all backup files and remove old ones as necessary.

preserve_size

Sets the maximum size, in Kilobytes (KB), of the log file when the RTF is run with preservation mode turned ON. The default setting is 300. When the file reaches its maximum size, RTF "rolls over" the log file, much like a circular buffer. The previous log file is saved as a timestamped text file. RTF aligns the entries so that the oldest entry is always at the top of the log file and the newest entry is always at the bottom of the log file.

- Note:** This attribute is only applicable when RTF preservation mode is turned ON. When RTF preservation mode is OFF, the size attribute determines the size of the log file. Refer to [Section 27.5.3, "Running RTF in Preservation Mode"](#), on page 138 for information about preservation mode.

preserve_maxbackups

Indicates the maximum number of backup log files the RTF creates when the RTF is run with preservation mode turned ON. If this attribute is set to 0, all trace information is written to one log file. If this attribute is set to 1 or greater, all trace information is initially written to one log file. When the size of this initial log file reaches the threshold defined in the Logfile tag's preserve_size attribute, the RTF trace data “rolls over” into a second log file. When the size threshold is reached in the second log file, the RTF trace data “rolls over” into a third log file. This sequence occurs until the number of backup log files created equals the preserve_maxbackups attribute setting. When the “roll over” occurs, the previous log file is saved under the following timestamped name:

rtflog-<year><month><day>-<hour>h<minute>m.<second>s_p.txt

Note: The _p suffix appears in the log file name because preservation mode is turned ON.

The default preserve_maxbackups value is 10.

Note: This attribute is only applicable when RTF preservation mode is turned ON. When RTF preservation mode is OFF, the maxbackups attribute determines the number of backup log files. Refer to [Section 27.5.3, “Running RTF in Preservation Mode”](#), on page 138 for information about preservation mode.

duplicate_to_debug_console

This attribute determines whether or not trace information is duplicated in the debug console (Windows) or the standard error output (Linux). Possible settings are as follows:

1

The trace output information will be duplicated on the debug console for Windows systems or the standard error output for Linux systems.

0

The trace output information will not be duplicated on the debug console for Windows systems or the standard error output for Linux systems. This is the default setting.

log_format

This attribute determines the format of the output log file. By default, this is set to “text” (the trace output log is written in readable, text file format). However, you can change the attribute to “binary” to get binary output.

27.3.3 Global Tag

The Global tag is the second child tag of the RTFConfig tag. The Global tag is used to specify the global configuration. Global configuration settings are valid for all modules included in the RTF configuration file. However, the Global tag has the lowest priority in the RTF configuration file. Therefore, the global configuration settings can be overridden by individual settings at the Module tag level (see [Section 27.3.7, “Module Tag”](#), on page 132). The Global tag cannot occur more than one time in the RTF configuration file. The Global tag can either be empty:

```
<Global>

<!-- This is an example of an empty Global tag -->

</Global>
```

or the Global tag can have GLabel and/or GClient child tags. There are no attributes associated with the Global tag.

27.3.4 GLabel Tag

The GLabel tag is a child tag of the Global tag; it is used to configure global labels. If a label is defined at the GLabel level then all module and client behavior will be governed by this configuration (unless overridden for a given module at the MLabel level).

The following line may appear in the default *RtfConfigLinux.xml* file:

```
<GLabel name = "Error" state = "1"/>
```

This line indicates that tracing of all Error labels is turned on by default. To disable this default behavior, you can delete this line or change the "Error" state attribute setting to 0.

The following lines appear in the default *RtfConfigWin.xml* file:

```
<GLabel name = "Error" state = "1"/>
<GLabel name = "Exception" state = "1"/>
```

These lines indicate that tracing of all Error labels and all Exception labels is turned on by default. To disable this default behavior, you can delete these lines or change the "Error" state attribute and "Exception" state attribute setting to 0.

The GLabel tag has the following two attributes:

name

Indicates the name of the global label to be configured. *You must define the name of the global label*; there is no default value. Examples of possible labels include:

- "Error" (enabled at the Global level by default)
- "Debug"
- "Info"
- "Warning"

Note: Refer to the default RTF configuration file's MLabel name attributes. These default entries are for the Intel Dialogic runtime libraries. Any of these runtime library label names can be included in a GLabel tag.

state

Specifies the state of the label. Valid values are as follows:

- 1
Label is enabled at the global level. All trace messages associated with this label will be sent to the trace output. This is the default value.
- 0
Label is disabled at the global level. Trace messages associated with this label will not be sent to the trace output.

27.3.5 GClient Tag

The GClient tag is a child tag of the Global tag; it is used to configure global clients (devices). If a client is defined at the GClient level then all client behavior will be governed by this configuration (unless overridden for a given client at the MClient level). The GClient tag can be empty or have GClientLabel children tags. The GClient tag has the following two attributes:

name

Indicates the name of the client to be configured. *You must define the name of the global client;* there is no default value. Examples of possible client names include:

- “dxxxB1C1” (voice device)
- “dxxxB2C2” (voice device)
- “ipmB2C2” (IP media device)

Note: Refer to the *Standard Runtime Library API for Windows Operating Systems Programming Guide* for more information about client names for Intel Dialogic libraries and devices.

state

Specifies the state of the client. Valid values are as follows:

- 1
Client is enabled at the global level. All trace messages associated with this client will be sent to the trace output. This is the default value.
- 0
Client is disabled at the global level. Trace messages associated with this client will not be sent to the trace output.

27.3.6 GClientLabel Tag

The GClientLabel tag is a child tag of the GClient tag; it is used to specify a label for a global client. The GClientLabel tag has the following two attributes:

name

Indicates the name of a client label to be configured. *You must define the name of the client label;* there is no default value. Examples of possible client labels include:

- “Error”
- “Warning”
- “Entry”

Note: Refer to the default RTF configuration file’s MLabel name attributes. These default entries are for the Intel Dialogic runtime libraries. Any of these runtime library label names can be included in a GClientLabel tag.

state

Specifies the state of the label. Valid values are as follows:

- 1
Client label is enabled at the global level. Trace messages associated with this label will be sent to the trace output. This is the default value.
- 0
client label is disabled at the global level. Trace messages associated with this label will not be sent to the trace output.

27.3.7 Module Tag

This tag is used to specify configuration for various modules. Module tags have a higher priority than global tags so settings at the module level override settings at the global level. For example, if the state of a label “Error” is set to “1” in the global section and “Error” is set to “0” for an individual module, then the label “Error” will not be traced for that particular module. The module section must exist in the configuration file, even if the section is empty. Possible child tags of the Module tag are MClient and MLabel.

The RTF configuration file contains modules for the Intel Dialogic runtime libraries. The following example shows that the module name for the Fax Library is “spwrfax”:

```
<!-- Fax Library -->
- <Module name="spwrfax" state="1">
  <MLabel name="DEBUG" state="0" />
  <MLabel name="INFO" state="0" />
  <MLabel name="APPL" state="0" />
  <MLabel name="WARN" state="0" />
  <MLabel name="ERR1" state="1" />
  <MLabel name="EXCE" state="0" />
</Module>
```

You can edit the state attributes of the modules to enable (set state = “1”) or disable (set state = “0”) the tracing. Alternatively, you can delete one or more modules from the default RTF configuration file if you are not interested in tracing certain runtime libraries.

Note: The RTF configuration file contains some older sections that have been retained for backward compatibility.

The Module tag includes the following attributes:

name

Indicates the name of a module to be configured. Intel Dialogic runtime libraries have module names in the default RTF configuration file. Example module names in the RTF configuration file include:

- “gc”
- “libsr1”
- “spwrdevmngmt”

state

Specifies the state of the module. Valid values are as follows:

- 1
Module is enabled. Trace messages associated with this label will be sent to the trace output. This is the default value.
- 0
Module is disabled. Trace messages associated with this label will not be sent to the trace output.

27.3.8 MLabel Tag

The MLabel tag is a child tag of the Module tag. The MLabel tag is used to configure module labels. If a label is defined at the global level and the same label is defined at the module level, the module level configuration overrides the global configuration for the module. The MLabel tag has the following two attributes:

name

Indicates the name of the label to be configured. The Intel Dialogic runtime libraries have module label names in the default RTF configuration file. Example module label names in the RTF configuration file include:

- “APPL”
- “DBG”
- “WARN”

state

Specifies the state of the label. Valid values are as follows:

- 1
Label is enabled. Trace messages associated with this label will be sent to the trace output. This is the default value.

0

Label is disabled. Trace messages associated with this label will not be sent to the trace output.

Note: When the state attribute is not included or not defined, the default value is 1. However, the Intel Dialogic runtime library module labels that exist in the default RTF configuration file have their state attribute initially set to 0.

27.3.9 MClient Tag

The MClient tag is a child tag of the Module tag; it is used to configure a specific client for the module. The MClient tag can be empty or have MClientLabel children tags. The MClient tag has the following two attributes:

name

Indicates the name of the client to be configured. The Intel Dialogic runtime libraries have pre-defined module client names in the default RTF configuration file. Example clients include:

- “dxxxB1C1” (voice device)
- “dxxxB2C2” (voice device)
- “ipmB2C2” (IP media device)

Note: Refer to the *Standard Runtime Library API for Windows Operating Systems Programming Guide* for more information about client names for Intel Dialogic libraries and devices.

state

Specifies the state of the client. Valid values are as follows:

1

Client is enabled. Trace messages associated with this client will be sent to the trace output. This is the default value.

0

Client is disabled. Trace messages associated with this client will not be sent to the trace output.

Note: When the **state** attribute is not included or not defined, the default value is 1.

27.3.10 MClientLabel Tag

The MClientLabel tag is a child tag of the MClient tag; it is used to specify a label for a client. The MClientLabel tag has the following two attributes

name

Indicates the name of a label to be configured. *You must define the name of the label*; there is no default value. Example labels include:

- “APPL”

- “DEBG”
- “WARN”

Note: A complete list of labels for a given library can be found in the RTF configuration file.

state

Specifies the state of the label. Valid values are as follows:

1

Label is enabled. Trace messages associated with this label will be sent to the trace output. This is the default value.

0

Label is disabled. Trace messages associated with this label will not be sent to the trace output.

27.3.11 Precedence Scheme for Module/Client Labels

This section summarizes the precedence scheme of settings in RTF module/client labels. Basically, RTF has three types of handles for use in determining if a trace statement will be filtered. They are modules, clients, and labels. A trace statement can use filtering of either of the following:

module+label

For trace statements that use module+label, the trace will only be stored in a file when both module and label are enabled.

module+client+label.

For trace statements that use module+client+label, the trace will only be stored in a file when module, client and label are all enabled.

When it is not provided in the configuration file, default enable/disable settings for the three types of handles are:

- Module: disabled.
- Client: enabled.
- Label: disabled.

For handles set in the configuration file, the precedence order is:

- Module: No precedence order since it is only available at a module level. No global level.
- Client: Settings in the module section override settings in the global section.
- Label: For trace statements *without* client handles, settings in the module section override settings in the global section. For trace statements *with* client handles, settings in the module/client section override settings in the client/label section; settings in the client/label section override settings in the module section; and settings in the module section override settings in the global section.

Figure 16 shows the hierarchy of the possible handles and how they relate to one another.

27.4 Restrictions and Limitations

Keep the following restrictions and limitations in mind when using RTF:

- If you run full RTF logging on high-density systems, you may experience I/O throughput degradation. It is recommended that you do not run RTF with full logging on high-density systems or any field-deployed systems. Instead, use just the default error-enabled logging.
- If you install the Intel Dialogic system release software on a FAT32 formatted hard drive, the RTF tool will generate an *Error after install attempting to launch* error message. The error message appears because FAT32 does not support file system access control. However, RTF will continue to function.
- The Intel Dialogic system software installation routine creates a *usr\dialogic\log* directory. When the system software is installed on a Mandriva* (formerly Mandrakesoft) Linux system, the Mandriva security utility (msec), which automatically runs every hour, may reset the access permissions to this *...log* directory. To avoid the resetting of permissions, you must perform the following before running RTF:
 1. Issue the `echo $SECURE_LEVEL` command. The Mandriva Linux system will return a number that indicates the current system security setting.
 2. Add the following line to the *perm.n* file (located in */usr/share/msec/perm.n*), where *n* indicates the security setting number returned in the previous step:

```
\var\log\dialogic\* current xxx
```

Where *xxx* indicates the desired access permission setting for the directory.
 3. After you change the *perm.n* file, run

```
msec n
```

so that the change will take effect immediately.
- When RTF logging is enabled, logs are stored in the *\$(INTEL_DIALOGIC_DIR)\log* directory. You should take necessary precautions to ensure that the directory never fills.
- Occasionally, errors that appear in the RTF log can be ignored in certain circumstances. Check the “Release Issues” chapter in the *Release Update* for the Intel telecom software you are using (System Release or Host Media Processing [HMP]). Such errors will be listed and explained in the “Release Issues” table.

27.4.1 Guidelines for Editing the RTF Configuration File

Keep the following rules in mind when editing the RTF configuration file:

1. Do not change the name of the file. The filename must remain at its default setting.
2. The RTF configuration file is broken down into four sections. The sequence of the sections must always be as follows:
 - a. RTFconfig
 - b. Logfile configuration
 - c. Global configuration
 - d. Module configuration
3. RTF uses the following default log file names:

- `rtflog-<year><month><day>-<hour>h<minute>m.<second>s.txt` when preservation mode is OFF.
 - `rtflog-<year><month><day>-<hour>h<minute>m.<second>s_p.txt` when preservation mode is turned ON.
4. The Global configuration section can only appear one time in the RTF configuration file.
 5. If there is configuration information which conflicts in the global and module sections (for example, the global section enables a trace label for a client, but the module section disables this same label for the same client), then the module configuration overrides the global configuration.
 6. Configuration information which is repeated later in the file will take precedence. For example, if there are two module configurations for *spwrvoice*, the configuration information lower in the file will dictate tracing behavior for that module.

Note: This should not be done, but it is mentioned here so the behavior can be recognized if this situation occurs by error.
 7. RTF is case sensitive. Therefore, the trace labels “Error” and “error” are considered to be two distinctly different trace labels.
 8. Simultaneous tracing of multiple Intel Dialogic libraries may have an adverse effect on system performance.
 9. If you run full RTF logging on high-density systems, you may experience I/O throughput degradation (specifically if you enable tracing on the MClient name MUTEX). It is recommended that you do not run RTF with full logging on high-density systems or any field-deployed systems. Contact customer support before enabling extensive logging. Instead, use just the default error-enabled logging.
 10. RTF affects running processes/applications only. If you enable RTF prior to starting your application, the RTF will not provide trace information until your application has started.
 11. If you wish to dynamically edit the RTF trace levels while your application runs, it is not necessary to stop RTF. Instead, perform the following:
 - a. Open the RTF configuration file.
 - b. Customize the settings in the RTF configuration file.
 - c. Save and close the RTF configuration file.
 - d. Issue the `rtftool reload` command to reload RTF configuration file and restart the RTF engine.

Note: Keep in mind that changes made to the RTF configuration file will not be reflected in the RTF output until the tool has been reloaded.
 12. At any time, you can issue the `rtftool` command to pause/restart RTF. RTF will not provide trace output while it is paused. Refer to [Section 27.5, “rtftool Command”](#), on page 138 for more information.
 13. Save the original *RtfConfigWin.xml* file or *RtfConfigLinux.xml* file, as it is advisable to return to the original logging level when finished using the RTF logging mechanism.

27.5 rtftool Command

This section explains the `rtftool` command, which is used to control RTF and modify the trace output. The `rtftool` command is issued from the command line. The `rtftool` command can be issued from any directory.

27.5.1 Pausing and Reloading RTF

RTF tracing is enabled by default. Use the `rtftool` command to pause, resume and reload the RTF's tracing capabilities. The `rtftool` command has the following pause and reload variations:

`rtftool pause`

Pauses RTF tracing. Trace output will not be written to the log file while RTF is paused.

`rtftool resume`

Resumes RTF tracing. Issue this command to resume tracing after RTF has been paused.

`rtftool reload`

Reloads an updated RTF configuration file (*RtfConfigLinux.xml* or *RtfConfigWin.xml*) and restarts RTF, using the settings from the updated RTF configuration file.

Note: You do not need to pause RTF tracing before issuing the `rtftool reload` command. RTF supports dynamic updates to the RTF configuration file. Simply edit and save the file, then issue the `rtftool reload` command to begin tracing with the updated RTF configuration file settings.

27.5.2 Clearing the RTF Trace Log File Contents

Use the `rtftool` command to clear all trace data from the RTF log file. The following command line option clears the RTF log file contents:

`rtftool clean [-f]`

Clears the RTF log file contents. When the `-f` option is used, RTF will not ask the user for confirmation before clearing the log file.

27.5.3 Running RTF in Preservation Mode

You can set RTF to run in preservation mode. Preservation mode allows you to determine whether old trace data is overwritten or preserved in a separate file when the log file grows to its maximum size. When preservation mode is turned ON, RTF will append a `_p` to the end of each RTF log file name; RTF will not delete or overwrite these `_p` log files. Use the `preserve_size` and `preserve_maxbackup` Logfile tag attributes to set the size and number of backup log files, respectively. Use the following command line option to enable/disable preservation mode:

`rtftool preservation {on | off}`

Turns RTF preservation mode ON and OFF.

27.6 Example RTF Configuration Files

This section provides a number of example *RtfConfig*.xml* files along with a brief explanation of how the file settings affect the RTF trace output. Note that the same rules/examples covered in this section apply to both the *RtfConfigWin.xml* and *RtfConfigLinux.xml* file.

27.6.1 Example 1: Tracing disabled - RtfConfigWin.xml

```
<?xml version="1.0" standalone="no"?>

<RTFConfig trace="0" tracelocation="TRACE_LOG" logformat="ALIGN"
timestamp="1">

<Logfile path="$(INTEL_DIALOGIC_DIR)\log" size="1000" maxbackups="2"
preserve_size="300" preserve_max_backups="10"
duplicate_to_debug_console="0" log_format="text"/>

<Global>

</Global>

<Module name="spwrvoice" state = "1">
    MLabel name="ERR1" state = "1"/>
    MLabel name="WARN" state = "0"/>
</Module>

</RTFConfig >
```

Explanation

The RTFConfig tag's trace attribute is set to 0 so tracing is disabled. Trace output will not be created. Set the trace attribute to 1 to activate tracing.

27.6.2 Example 2: Tracing enabled, logfile path and size specified, preservation mode OFF, one module configured - RtfConfigLinux.xml

```
<?xml version="1.0" standalone="no"?>

<RTFConfig trace="1" tracelocation="TRACE_LOG" timestamp="1"
logformat="ALIGN">

<Logfile path="$(INTEL_DIALOGIC_DIR)/log" size="1000" maxbackups="2"
preserve_size="300" preserve_max_backups="10"
duplicate_to_debug_console="0" log_format="text"/>
```

```

<Global>

</Global>

<Module name="spwrvoice" state = "1">
    MLabel name="ERR1" state = "1"/>
    MLabel name="WARN" state = "1"/>
</Module>

</RTFConfig >

```

Explanation

The RTFConfig tag's trace attribute is set to 1 so tracing is enabled. The Global tag is empty, so there is no global configuration. For this example, tracing is only configured at the module level.

The path for the log file is \$(INTEL_DIALOGIC_DIR)/log. Preservation mode is OFF, so the size and maxbackup attributes apply while the preserve_size and preserve_max_backups attributes are ignored. The maximum logfile size is 1000KB. The system maintains a maximum of 2 backup log files.

The only module configured for trace is *spwrvoice*. This means that all clients (devices) for this module are configured to trace ERR1 and WARN labels.

27.6.3 Example 3: Tracing enabled, logfile path and size specified, several modules configured, global configuration used - RTFConfigWin.xml

```

<?xml version="1.0" standalone="no"?>
<RTFConfig trace="1" tracelocation="TRACE_LOG" logformat="ALIGN">

<Logfile path="$(INTEL_DIALOGIC_DIR)\log" size="1000" maxbackups="2"
preserve_size="300" preserve_max_backups="10"
duplicate_to_debug_console="0" log_format="text"/>

<Global>
    <GLabel name="Entry" state = "1"/>
    <GClient name="dxxxBlC2" >
        <GClientLabel name="Exit" state ="1"/>
    </GClient>
</Global>

<Module name="spwrvoice" state = "1">
    MLabel name="ERR1" state = "1"/>
    MLabel name="WARN" state = "0"/>
</Module>

```

```

<Module name="spwrvoice">
  <MLabel name="ERR1" state = "0"/>
  <MLabel name="WARN" state = "1"/>

  <MClient name="dxxxB1C2" >
    <MClientLabel name="ERR1"/>
  /MClient>
</Module>

<Module name="faxntf" state = "1">
  <MLabel name="WARN" state = "1"/>
</Module>

<Module name="dtintf">
  <MClient name="dxxxB2C1">
    <MClientLabel name="EXCE"/>
  </MClient>

</Module>

</RTFConfig >

```

Explanation

The trace attribute of the RTFConfig tag is set to 1 so tracing is enabled.

The logfile path is $\$(INTEL_DIALOGIC_DIR)\log$. Preservation mode is ON, so the size and maxbackup attributes are ignored while the preserve_size and preserve_max_backups attributes apply. The maximum logfile size is 300KB. The system maintains a maximum of 10 backup log files (filenames appended with a _p).

The Global section is present in the configuration file, so all modules will have this global configuration. This configures “Entry” as a global label and “dxxxB1C2” is a global client for label “Exit”.

In the module section, there are two configurations for *spwrvoice* so the later configuration will take precedence. Since the state of *spwrvoice* is not specified, it takes default value “1”. Tracing is enabled for the module *spwrvoice* but only for the “WARN” label.

The client “dxxxB1C2” is also configured to trace in the *spwrvoice* module with the label “ERR1”. Therefore the client “dxxxB1C2” in *spwrvoice* is traced for the label “ERR1”, even though it is disabled in the module section. While all other clients of *spwrvoice* are configured to be traced for only label “WARN”, as “WARN” is the only label configured to be traced for the module.

The *faxntf* module is configured to trace “WARN” (from the module section) and “Entry” (from the global section).

The *dtintf* module is configured to trace “Entry” (from the global section). “dxxxB1C2” in *dtintf* is configured to trace “Entry” (from the global section) and “Exit” (from the global client). The other client “dxxxB2C1” is configured to trace “Entry” (from the global configuration) and “EXCE” (from the module client section). All other clients of this module are configured to trace “Entry” (from the global section) since these labels are configured for the module.

27.7 Configuring Remote RTF Logging

RTF servers can be configured to communicate with each other. This provides remote tracing capability. A trace produced on one machine can be re-directed and stored on another machine on the network. This section describes how to configure remote RTF logging.

The terminology used to describe remote RTF logging is as follows:

Client (remote) system

A system that contains System Release software and telephony boards and is running the end-user telephony application.

Server (management/help desk system)

A system that contains the System Release software and will be used to collect RTF logs from one or more client systems.

27.7.1 Configuring the Client System

To configure the client system, edit the `${INTEL_DIALOGIC_CFG}/RtfConfigLinux.xml` file as follows:

1. Locate the line containing the tag `Network mode=` and set its value to `"client"`.
2. Locate the line containing the tag `server_name=` and set its value to either the fully-qualified domain name or IP address of the server system.
3. Optionally, you may change the TCP port used by locating the line containing `Client port_number=` and changing its value to the port number you wish to use. Note that you must make a corresponding change on the server system (see [Configuring the Server System](#)).
4. If the System Release has not yet been started, run `dlstart`. Otherwise, cause the RTF program to reload the updated configuration file by running `rtftool reload`.

Figure 17 shows a portion of a sample *RtfConfigLinux.xml* file edited to configure the client system.

Figure 17. RTF Configuration File Edited to Configure the Client System

```
<?xml version="1.0" standalone="no" ?>
- <RTFConfig trace="1" tracelocation="TRACE_LOG" timestamp="1" logformat="ALIGN">
  <Logfile path="${INTEL_DIALOGIC_DIR}/log" size="500" maxbackups="10" preserve_size="500"
    preserve_maxbackups="10" duplicate_to_debug_console="0" log_format="text" />
  <StatusTrace state="off" />
  <Network mode="client" />
  <Server port_number="4567" max_connection="1" />
  <Client port_number="1234" server_name="192.168.93.227" />
```

27.7.2 Configuring the Server System

To configure the server system, edit the `${INTEL_DIALOGIC_CFG}/RtfConfigLinux.xml` file as follows:

1. Locate the line containing the tag `Network mode=` and set its value to "server". The value of `server_name=` must be 127.0.0.1.
2. Optionally, you may change the TCP port used by locating the line containing `Server port_number=` and changing its value to the port number you wish to use. Note that you must make a corresponding change on the client system(s) (see [Configuring the Client System](#)).
3. If the System Release has not yet been started, run `dlstart`. Otherwise, cause the RTF program to reload the updated configuration file by running `rtftool reload`.

Figure 18 shows a portion of a sample *RtfConfigLinux.xml* file edited to configure the server system.

Figure 18. RTF Configuration File Edited to Configure the Server System

```
<?xml version="1.0" standalone="no" ?>
- <RTFConfig trace="1" tracelocation="TRACE_LOG" timestamp="1" logformat="ALIGN">
  <Logfile path="${INTEL_DIALOGIC_DIR}/log" size="500" maxbackups="10" preserve_size="500"
    preserve_maxbackups="10" duplicate_to_debug_console="0" log_format="text" />
  <StatusTrace state="off" />
  <Network mode="server" />
  <Server port_number="1234" max_connection="1" />
  <Client port_number="4567" server_name="127.0.0.1" />
```

27.7.3 Cautions

Following are some cautions to keep in mind when configuring remote RTF logging:

1. If the value of the `server_name` field is a hostname, the name must be resolvable either via DNS query or via `/etc/hosts`. Failure of hostname resolution will prevent remote logging from occurring.
2. Depending on the level of RTF logging enabled, application or network performance may be impacted. The impact may be more significant if VoIP technology is being used.
3. Having multiple client systems log to a central server is possible by setting the `max_connection=` parameter in the server's RTF configuration file, but this capability should be used with caution.

This chapter provides reference information about the Status Monitor tool. Use of the Status Monitor tool differs for Windows and Linux operating systems, so the tool is described in the following sections:

- [Status Monitor for Windows](#) 145
- [Status Monitor for Linux](#) 148

28.1 Status Monitor for Windows

This section describes the Windows version of the tool and includes the following information:

- [Description](#)
- [Guidelines](#)
- [Options](#)

28.1.1 Description

The Status Monitor tool enables you to track the state of the TSC as well as the state of the bits on a robbed bit or CAS line. You can use Status Monitor for both troubleshooting and administration. For troubleshooting, you can monitor the call state for problems such as hung channels. For administration, you can watch call progress and channel usage.

28.1.2 Guidelines

Following are guidelines for using the Status Monitor (for Windows) tool:

- The Status Monitor 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 take a long time to start the monitoring: about 1 minute per board.
- The CAS instance is not valid until the line is in service, so you will have to run an application or the Lineadmin and Phone tools to set the channel in-service before using the bit monitoring feature. For information about the Lineadmin tool, refer to the Administration Guide for the system release. For information about the Phone tool, refer to [Chapter 23, “Phone Reference”](#).
- The Status Monitor tool is best viewed in resolutions greater than 1024 x 768.
- The Status Monitor tool is not intended for use with boards that are configured for ISDN.

28.1.3 Options

Status Monitor has the following option:

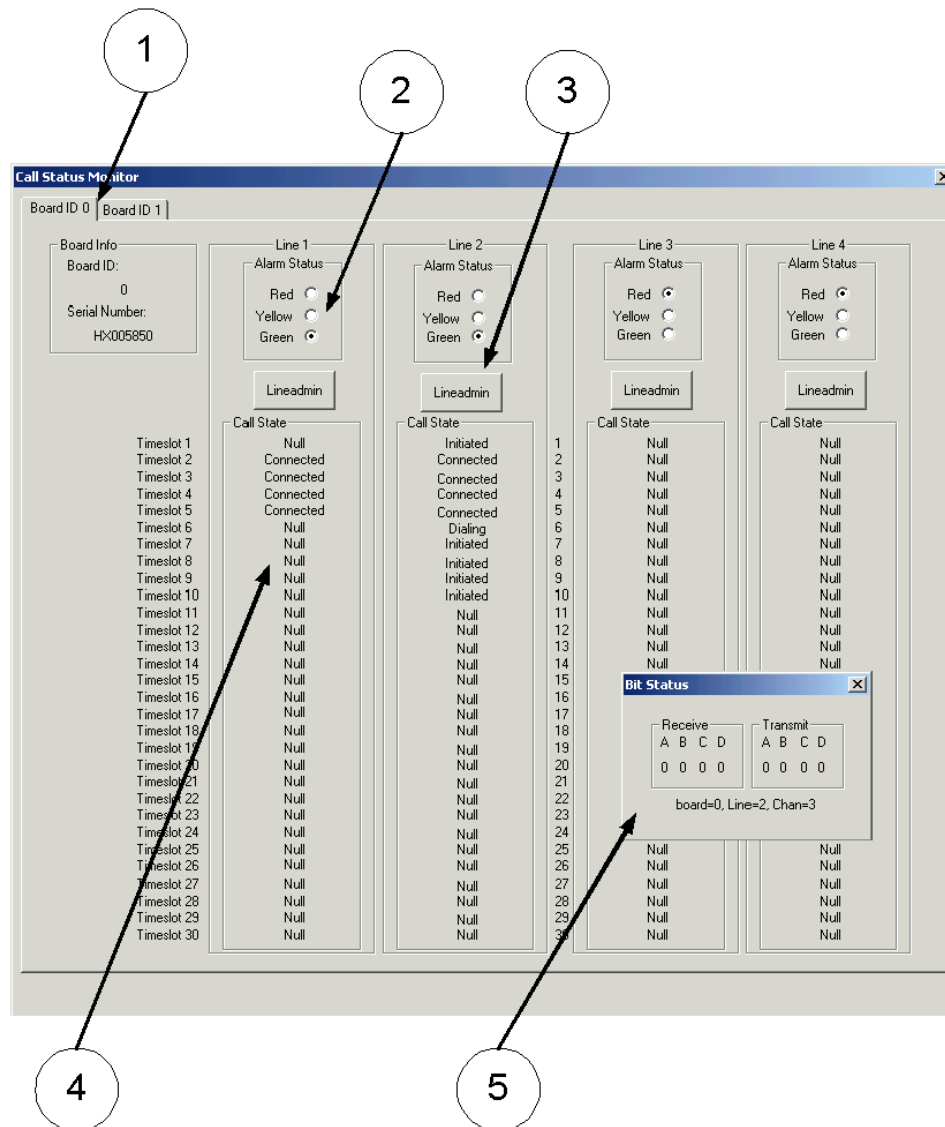
-board <board list>

Logical ID of the board(s) to trace (optional). The default is to monitor all DM3 boards in the system. Use the Configuration Manager (DCM) utility to obtain the board's logical ID.

The following will run the Status Monitor tool on boards 0 and 1:

```
StatusMon -board 0 1
```

Figure 19 provides an example of output from the Status Monitor tool.

Figure 19. Example of Status Monitor (for Windows) Output


The following numbered list corresponds to the labels in Figure 19.

1. You can toggle between monitored boards by clicking on the tab that represents the board ID.
2. This part of the display shows the current alarm state on the trunk (red, yellow, or green).
3. Use this button to invoke the Lineadmin tool to view and set additional alarms.
4. This part of the display allows you to monitor the call state for each line on the board.
5. Clicking on a line will cause the bit information to be displayed in this window. (All 1's will be displayed for ISDN lines.)

28.2 Status Monitor for Linux

This section describes the Linux version of the tool and includes the following information:

- [Description](#)
- [Guidelines](#)
- [Usage](#)

28.2.1 Description

The Linux equivalent of Status Monitor is the statusmon application. It is a command line application with a terminal user interface (TUI). The application supports two execution modes:

- monitoring call state
- monitoring bit information.

The call state monitoring mode corresponds to the StatusMon (Windows version) initial page. However, the Linux version does not support tabs to switch between boards. The bit information monitoring mode is equivalent to the window that appears when clicking on a single channel in the StatusMon (Windows version) display. You can launch multiple instances of statusmon for concurrent monitoring in either mode.

28.2.2 Guidelines

This section provides the following guidelines for using the statusmon (for Linux) tool:

- The statusmon application will only function when the Intel Dialogic system is running and at least one board is in the DOWNLOADED state. The behavior of statusmon is undefined if the Intel Dialogic system is stopped while statusmon is running. In such a case, the statusmon application should be exited and restarted.
- The statusmon application only works with Intel NetStructure® on DM3 architecture boards that are configured for CAS. The statusmon tool is not intended for use with boards that are configured for ISDN.
- The statusmon application does not validate the configuration (validate use of CAS).

28.2.3 Usage

This section describes how to use the statusmon (for Linux) application and shows sample screens.

Usage is as follows:

```
statusmon board  
or  
statusmon board trunk channel
```

where **board** is the logical ID of the board to monitor. Use the listboards utility to obtain the board's logical ID.

Note: The listboards utility is described in the Administration Guide for the release.

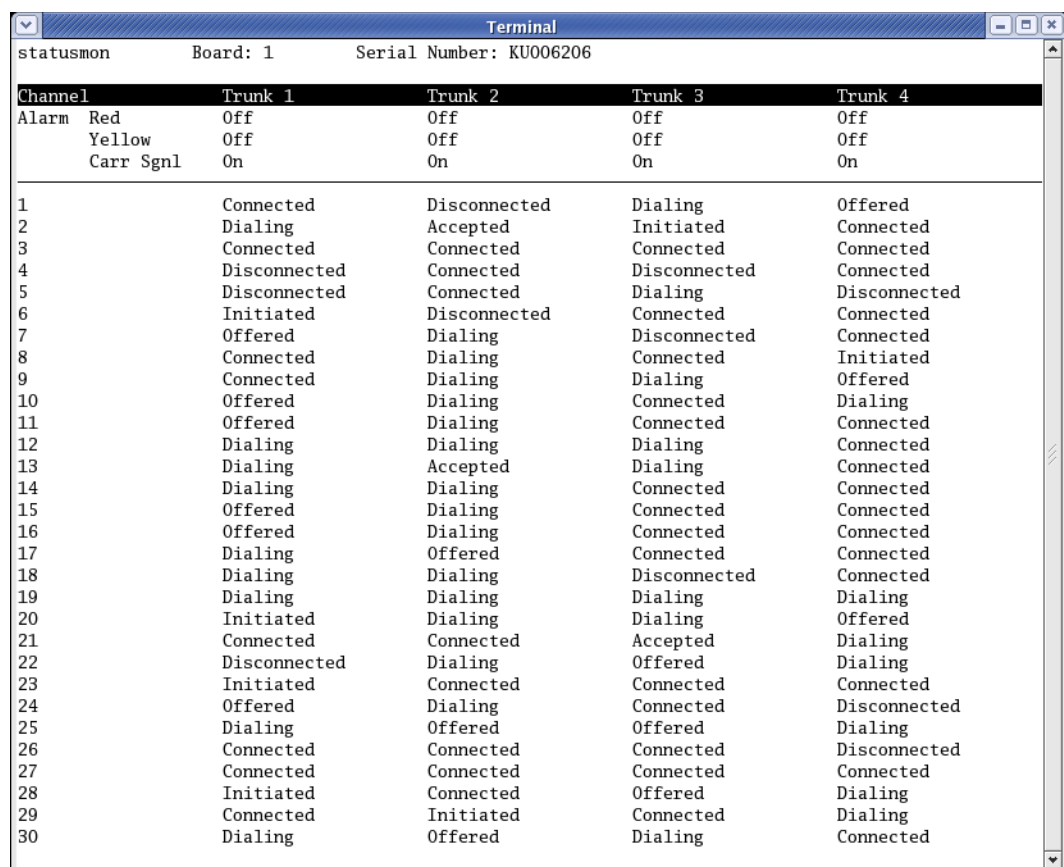
To exit the statusmon application gracefully, press the **q** key.

The statusmon application can be used in the following modes:

- Monitor the call state for each trunk of the board
statusmon board
- Monitor bit information on the specified channel
statusmon board trunk channel

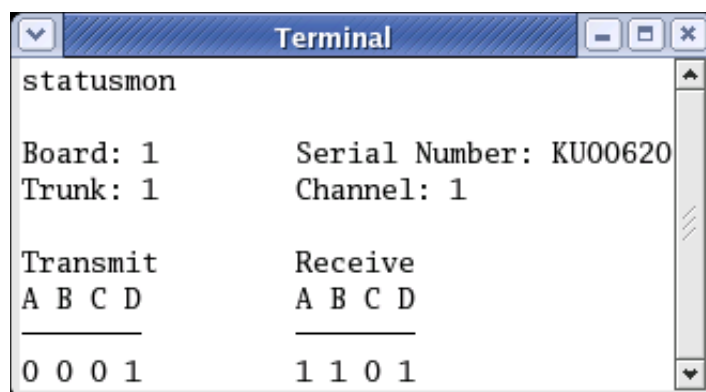
Following is an example of each mode:

Figure 20. Statusmon (for Linux) Call Mode



Terminal					
statusmon Board: 1 Serial Number: KU006206					
Channel		Trunk 1	Trunk 2	Trunk 3	Trunk 4
Alarm	Red	Off	Off	Off	Off
	Yellow	Off	Off	Off	Off
	Carr Sgnl	On	On	On	On
1		Connected	Disconnected	Dialing	Offered
2		Dialing	Accepted	Initiated	Connected
3		Connected	Connected	Connected	Connected
4		Disconnected	Connected	Disconnected	Connected
5		Disconnected	Connected	Dialing	Disconnected
6		Initiated	Disconnected	Connected	Connected
7		Offered	Dialing	Disconnected	Connected
8		Connected	Dialing	Connected	Initiated
9		Connected	Dialing	Dialing	Offered
10		Offered	Dialing	Connected	Dialing
11		Offered	Dialing	Connected	Connected
12		Dialing	Dialing	Dialing	Connected
13		Dialing	Accepted	Dialing	Connected
14		Dialing	Dialing	Connected	Connected
15		Offered	Dialing	Connected	Connected
16		Offered	Dialing	Connected	Connected
17		Dialing	Offered	Connected	Connected
18		Dialing	Dialing	Disconnected	Connected
19		Dialing	Dialing	Dialing	Dialing
20		Initiated	Dialing	Dialing	Offered
21		Connected	Connected	Accepted	Dialing
22		Disconnected	Dialing	Offered	Dialing
23		Initiated	Connected	Connected	Connected
24		Offered	Dialing	Connected	Disconnected
25		Dialing	Offered	Offered	Dialing
26		Connected	Connected	Connected	Disconnected
27		Connected	Connected	Connected	Connected
28		Initiated	Connected	Offered	Dialing
29		Connected	Initiated	Connected	Dialing
30		Dialing	Offered	Dialing	Connected

Figure 21. Statusmon (for Linux) Bit Mode



```
statusmon

Board: 1          Serial Number: KU00620
Trunk: 1          Channel: 1

Transmit          Receive
A B C D          A B C D
-----          -----
0 0 0 1          1 1 0 1
```

This chapter provides reference information about the UDD diagnostics tool, which is used with Intel® Dialogic® Springware architecture boards. Procedures for using UDD are provided in [Chapter 4, “Checking Springware Architecture Boards”](#). This reference chapter contains the following topics:

- [Description](#) 151
- [User Interface Considerations](#) 151
- [Error Messages from UDD Startup](#) 152
- [UDD GUI Screens](#) 153
- [UDD Tests](#) 155

29.1 Description

UDD is a diagnostic utility used for testing the functionality of Intel Dialogic boards with Springware architecture. The utility relies on configuration data used to download firmware to a board. Any Springware architecture board configured to operate in a system and supported by the current version of UDD will appear on the UDD Main screen.

Note: For Linux only, a Java Runtime Environment is required for operation of UDD.

UDD contains a list of tests that can be selected, deselected, and executed a specified number of times on each board. The tests available are specific to each type of board. For example, if you select the D/240SC-T1 board from the Main screen, only the tests specific to that board will appear on the Specify Tests screen.

29.2 User Interface Considerations

Selections are made using the mouse. In a Windows environment, shortcut keys can also be used by pressing the **Alt** key and the underlined letter of the screen selection. For example, “**R**un Tests” would be **Alt-R**. For more shortcut keys, refer to [Section 29.4.3, “UDD Main Screen Button Options”](#), on page 154 and [Section 29.4.4, “UDD Specify Tests Screen Buttons”](#), on page 154.

In Linux, the **Alt** key is not a function, but may be simulated by pressing the **Esc** key. For example, to simulate **Alt-X**, press the **Esc** key and then the **X** key. To exit, press the **Esc** key twice.

- Notes:**
1. **For Windows:** Online help is available whenever the **F1** key is pressed. Pressing **F1** once will retrieve specific help for the current screen choice. Pressing **F1** again will retrieve more general help.
 2. **For Linux:** Online help is available using the mouse.

29.3 Error Messages from UDD Startup

An error message will appear if a problem prevents the UDD utility from starting. Possible error messages are listed in Table 2.

Note: Unless indicated otherwise, UDD will exit immediately if one or more of the messages listed in Table 2 is returned.

Table 2. Error Messages

Message Returned	Possible Cause/Solution
No supported or downloadable boards found in current configuration.	No boards in the system. Configuration data exists, but was not found or failed to download.
Cannot determine product for board pair: board ID = <number>, daughter ID = <number>	The board product IDs read from the EEPROMs do not match a known or supported product. This may indicate a board failure. Contact Intel's Telecom Support Resources (http://www.intel.com/design/telecom/support/). If any board in the system does not have this problem, UDD will continue.
Cannot send command to <board name>. Cannot read EEPROM for <board name>.	Unable to communicate with the board. Check the hardware and software configuration. If any board in the system does not have this problem, UDD will continue.
Cannot find file: <filename>.	A required file was not found. Verify that all required packages are installed.
Environment variable <variable name> is not set.	The environment variables DLCFGPATH and DLFWLPATH are used by UDD to locate files. These variables are often set in <i>CONFIG.SYS</i> when you install other required Intel Dialogic software. To verify that they are currently set, type "set" to list variables and their values. Note: Environment variables will be changing with the next full release of the Intel Dialogic software. See the Overview section of the Software Installation Guide for details.
Cannot access board. Skipping these boards. Boards must be stopped before running this program. Press Enter to continue.	An application is running or the system cannot communicate with the board. Stop the boards before starting UDD.
Cannot open: <configuration data>	Incorrect <i>DIALOGIC.CFG</i> file or registry entry. Unable to open the configuration data.
<configuration data>: invalid line: <line #>	Check the configuration data.
Unknown board type specified for <board>	Unknown/unsupported board. If any board in the system does not have this problem, UDD will continue.
Note: The messages shown in this table will appear on the screen, but will not be listed in the log file.	

Table 2. Error Messages (Continued)

Message Returned	Possible Cause/Solution
Download of system failed followed by one of the following: Fatal error opening boot file. Error reading boot file. Unable to open GENBOOT.FWL. Unable to open firmware file. Old format or corrupted firmware file. GENBOOT.BIN is version 1.10 or earlier. GENBOOT not responding on board xx. GENBOOT failed on board xx. Failed to initiate GENBOOT download. Failed to start firmware download of diagnostics firmware.	Messages of this type indicate the following: A corrupt file or nonexistent file. GENBOOT version is too old. No communication with product.
No supported or downloadable boards found in current configuration data. UDD Exiting.	The system cannot find boards. Check configuration data.
Note: The messages shown in this table will appear on the screen, but will not be listed in the log file.	

29.4 UDD GUI Screens

This section provides the following information about the UDD GUI Screens:

- [Board Information Displayed on UDD Main Screen](#)
- [UDD Main Screen Menu Options](#)
- [UDD Main Screen Button Options](#)
- [UDD Specify Tests Screen Buttons](#)

29.4.1 Board Information Displayed on UDD Main Screen

The Main screen displays the following information about each board:

Product	Intel Dialogic product name.
Board Name	The name given to the board in configuration data, ID=xx, where xx is the board number or address. Note that baseboard, daughterboard, dual board, and dual daughterboard have their own ID.
Serial No.	Board serial number.
Addr	Base shared RAM address.
Len	Length of shared RAM.
IRQ	Valid hardware interrupt level.
ID	Board locator number. H represents hexadecimal (daughterboard id = baseboard id + 20Hex). For D/41D boards, the board ID is the base shared RAM address.

29.4.2 UDD Main Screen Menu Options

The UDD Main screen contains the following menu selections:

Menu Item	Action
File -> Exit	Exit the UDD utility
Options -> Log Test Results	Create a log file for test results
Help -> UDD Help	Display online help for the Main screen
Help -> About	Displays the UDD version and copyright information

29.4.3 UDD Main Screen Button Options

The UDD Main screen consists of the buttons shown in Table 3. Use the mouse to navigate the Main screen.

Table 3. UDD Main Screen Buttons

Button	Action	Windows Shortcut Key
Select All	Selects all boards listed on the Main screen.	Alt-A
Deselect All	Deselects any boards currently selected.	Alt-D
Load Test Set	Not supported.	Alt-L
Save Test Set	Not supported.	Alt-S
Specify Tests	Specifies which tests to run by displaying a Test Selection screen (one for each board selected).	Alt-T
View Log File	Displays a log file of test results if logging was enabled. For more information, refer to Section 4.6, "Generating a Log File" , on page 34.	Alt-V
Run Tests	Executes all selected tests.	Alt-R
Exit UDD	Exits the UDD utility. Wait for UDD to terminate completely before proceeding with an application.	Alt-X

29.4.4 UDD Specify Tests Screen Buttons

The buttons available on the Specify Tests screen are listed in Table 4.

Table 4. Specify Tests Buttons

Menu Option	Action	Windows Shortcut Key
Select <u>A</u> ll Tests	Selects all tests specific to a selected board. There is an option to set test parameters for each test. Test parameters are set by selecting either the “Use Default Parameters” or “Use Custom Parameters” column from the Specify Tests screen.	Alt-A
<u>D</u> eselect All Tests	Deselects any tests currently selected. This option does not reset any parameters.	Alt-D
<u>O</u> K	Returns to the Main screen after tests have been selected for a specific board.	Alt-O
<u>C</u> ancel	Returns to the Main screen before all tests have been selected.	Alt-C
<u>H</u> elp	Activates context sensitive online help.	Alt-H

29.5 UDD Tests

This section describes all the tests you can perform on Intel Dialogic Springware architecture boards using UDD. Procedures for performing these tests are provided in [Chapter 4, “Checking Springware Architecture Boards”](#). This section contains information about the following tests:

- [Board Sanity Check](#)
- [Shared RAM Test](#)
- [Local RAM Test](#)
- [DSP Test](#)
- [Board Timer Test](#)
- [Onboard Interrupt Test](#)
- [SC2000 Test](#)
- [EEPROM Test](#)
- [PC Interrupt Test](#)
- [T1 Loopback Test](#)
- [E1 Loopback Test](#)
- [LS Frontend Test](#)
- [NS Hardware Test](#)

29.5.1 Board Sanity Check

The Board Sanity Check tests the basic functionality of the board. This test includes short versions of the Shared RAM, Local RAM, DSP, EEPROM, and Board Timer tests.

Parameters

Number of Iterations	Decimal number indicating the number of times the test will be executed. Default: 1 Valid Range: 1 to 32767
----------------------	-------------------------------------------------------------------------------------------------------------------

29.5.2 Shared RAM Test

The Shared RAM Test verifies the available memory shared between the host and the board selected.

Parameters

Number of Iterations	Decimal number indicating the number of times the test will be executed. Default: 1 Valid Range: 1 to 32767
----------------------	-------------------------------------------------------------------------------------------------------------------

29.5.3 Local RAM Test

The Local RAM Test checks the local memory of the control processor on the selected board.

Parameters

Number of Iterations	Decimal number indicating the number of times the test will be executed. Default: 1 Valid Range: 1 to 32767
Starting Address	Zero-based hexadecimal value of the address in Local RAM where the memory test will be started. Default: minimum offset allowed (0x10000 on D/xxxSC) Valid Range: 0x10000 to 0xDFFFF
Number of Bytes	Hexadecimal number defining the number of consecutive bytes in tested memory, starting from Starting Address. Default: 0xD0000 Valid Range: 1 to 0xD0000

29.5.4 DSP Test

The DSP Test verifies the board's Digital Signal Processing (DSP) subsystem. This test checks DSP memory, interrupts between the DSP and the control processor, and the DSP's crystal speed rate.

Parameters

Number of Iterations	Decimal number indicating the number of times the test will be executed. Default: 1 Valid Range: 1 to 32767
Select DSP	Check box to select DSP X and/or DSP Y. If the board has only one DSP, the DSP X box is checked and both DSPs are greyed out.

29.5.5 Board Timer Test

The Board Timer Test checks the functionality of the control processor's Timer 2 channel on the selected board.

Parameters

Number of Iterations	Decimal number indicating the number of times the test will be executed. Default: 1 Valid Range: 1 to 32767
----------------------	-------------------------------------------------------------------------------------------------------------------

29.5.6 Onboard Interrupt Test

The Onboard Interrupt Test checks the functionality of the local interrupt system on the selected board.

Parameters

Number of Iterations	Decimal number indicating the number of times the test will be executed. Default: 1 Valid Range: 1 to 32767
----------------------	-------------------------------------------------------------------------------------------------------------------

29.5.7 SC2000 Test

The SC2000 Test applies only to boards having SC2000 circuitry. SC2000 circuitry supports the SCbus, which is the hardware pathway used for communication among Intel Dialogic boards. The SC2000 test verifies SC2000 circuitry by performing a data loopback test, where data is transmitted from a DSP, looped back through the SC2000, and validated by the DSP.

Parameters

Number of Iterations	Decimal number indicating the number of times the test will be executed. Default: 1 Valid Range: 1 to 32767
----------------------	-------------------------------------------------------------------------------------------------------------------

29.5.8 EEPROM Test

The EEPROM Test verifies the selected board's ability to read the Electronically Erasable Programmable Read Only Memory (EEPROM). A board uses this memory to store configuration data.

Parameters

Number of Iterations	Decimal number indicating the number of times the test will be executed. Default: 1 Valid Range: 1 to 32767
----------------------	-------------------------------------------------------------------------------------------------------------------

29.5.9 PC Interrupt Test

The PC Interrupt Test checks the board's ability to generate interrupts to the PC. As the board generates interrupts, the host counts them and verifies that the correct amount has been generated.

Parameters

Number of Iterations	Decimal number indicating the number of times the test will be executed. Default: 1 Valid Range: 1 to 32767
----------------------	-------------------------------------------------------------------------------------------------------------------

29.5.10 T1 Loopback Test

The T1 Loopback Test applies to boards having a T-1 front end. This test uses the front end set to local loopback on boards with a T-1 digital network interface to perform signaling and data loopback tests. It is a stand-alone test and does not require any other network interface boards or external loopback connector.

Parameters

Number of Iterations	Decimal number indicating the number of times the test will be executed. Default: 1 Valid Range: 1 to 32767
Signaling Test	The DSP generates signaling information and verifies that it receives the same signaling information. The signaling test is run four times, alternating the possible patterns for the A and B signaling bits (for example, AB = 00, 01, 10, 11). Default: run test on all time slots Valid Range: 1 to 24
Data Test	Generates a data pattern in the DSP and then verifies that the DSP received the correct data. Default: run test on all time slots Valid Range: 1 to 24

29.5.11 E1 Loopback Test

The E1 Loopback Test applies to boards having an E-1 front end. This test uses the front end set to local loopback on boards with an E-1 digital network interface to perform signaling and data loopback tests. It is a stand-alone test and does not require any other network interface boards or external loopback connector.

Parameters

Number of Iterations	Decimal number indicating the number of times the test will be executed. Default: 1 Valid Range: 1 to 32767
Signaling Test	The DSP generates signaling information and verifies that it receives the same signaling information. The signaling test is run twice. Default: run test on all time slots Valid Range: 1 to 32
Data Test	Generates a data pattern in the DSP and then verifies that DSP received the correct data. Default: run test on all time slots Valid Range: 1 to 32

29.5.12 LS Frontend Test

The LS Frontend Test applies to boards having a loop start front end. The LS Frontend default parameters test Dual Subscriber Line Audio-processing Circuits (DSLAC) by programming each DSLAC channel and verifying that the correct information was written. Tones are also played and verified on each DSP channel. This is a stand-alone test and does not require other network interface boards or an external loopback connector.

Parameters

Number of Iterations	Decimal number indicating the number of times the test will be executed. Default: 1 Valid Range: 1 to 32767
Analog Test	Tests the DSLACs in analog loopback mode. Default: run the test on all channels Valid Range: 1 to 16
Digital Test	Tests the DSLACs in time slot assigned loopback mode. Default: run the test on all channels Valid Range: 1 to 16

29.5.13 NS Hardware Test

The NS Hardware Test applies to boards having 8031/80188 interrupts and an XILINX chip. The NS Hardware test verifies the Northern Telecom interface on the NS daughterboard of the D/42-NS board. It first tests the 8031/80188 mailbox and shared RAM, and then tests the C-84 chip set, which interfaces the NS daughterboard to the PBX.

Parameters

Number of Iterations	Decimal number indicating the number of times the test will be executed. Default: 1 Valid Range: 1 to 32767
----------------------	-------------------------------------------------------------------------------------------------------------------

Glossary

ANI: Automatic Number Identification. A telephone service that provides the telephone number of an incoming call. ANI is also known as Caller ID.

Automatic Number Identification: See ANI.

B channel: A bearer channel that carries the main data.

bit oriented : Communications protocols in which control information may be coded in a single bit.

CAS: Channel Associated Signaling. Signaling in which the signals necessary to switch a given circuit are transmitted via the circuit itself or via a signaling channel permanently associated with it.

called ID: A telephone service that identifies for the receiver what telephone number was dialed by the caller. This is also known as DNIS.

caller ID: A telephone service that provides the telephone number of an incoming call. Caller ID is also known as ANI.

channel: A path of communication, either electrical or electromagnetic, between two or more points. Also called a circuit, facility, line, link or path.

D channel: A channel that carries control and signaling information.

Dialed Number Identification Service: See DNIS.

DNIS: Dialed Number Identification Service. A telephone service that identifies for the receiver what telephone number was dialed by the caller. This is also known as Called ID.

Integrated Services Digital Network: See ISDN.

ISDN: Integrated Services Digital Network. An integrated digital network in which the same time-division switches and digital transmission paths are used to establish connections for different services. ISDN services include telephone, data, electronic mail, and facsimile. The method used to accomplish a connection is often specified: for example, switched connection, nonswitched connection, exchange connection, ISDN connection.

line loopback: A signal used to command the far-end receiver to loop back the received line signal.

loopback: A state of a transmission facility in which the received signal is returned towards the sender; a type of diagnostic test in which the transmitted signal is returned to the sending device after passing through a data communications link or network, which allows the returned signal to be compared with the transmitted signal for troubleshooting purposes.

loop start: Seizing (starting) a line by bridging through a resistance the tip and ring (both wires) of a telephone line. In residential installations, the loop start trunk is the most common.

metallic circuit: A circuit in which metallic conductors are used and in which the ground or earth forms no part.

metallic loopback: Metallic loopback loops the data back to the network at the physical port on the back card of a service module, whereas local loopback loops the data back to the network through the framer in the service module.

OSD: Operating System Distribution.

payload: In a set of data, such as a data field, block, or stream, being processed or transported, the part that represents user information and user overhead information, and may include user-requested additional information, such as network management and accounting information.

payload loopback: A signal used to command the far-end receiver to loop back the received payload.

POST: Power-On Self Test. A series of diagnostic tests used to verify that the DM3 architecture board components are working properly.

PSTN: Public Switched Telephone Network.

Public Switched Telephone Network: PSTN. A domestic telecommunications network usually accessed by telephones, key telephone systems, private branch exchange trunks, and data arrangements.

signaling : To set up and tear down calls, some form of signaling is required (simple examples are ringing and dial tone).

trunk: In a communications network, a single transmission channel between two points that are switching centers or nodes, or both.

wink start: A signal sent between two telecommunications devices as part of a “handshake” protocol.

A

- aborting tests 33
- ALIGNED 126
- APPL 120
- application failure or crash 48
- Application Monitor tool 51
- AppMon 51
- AppMon.com 51
- AppMon.exe 51
- archiving system configuration 49

B

- Binary 117
- board information 153
- board sanity check 155
- board timer test 157
- board, checking 25
- board, processor or SRAM fault on 27

C

- CallInfo tool 53
- CAS Trace tool 55
- check network connection 35
- checking an individual DM3 architecture board 25
- checking the protocol configuration 39
- client 124
- Client Width 118
- ClientWidth 126
- collecting the system data 48
- command line options
 - pstndiag tool 109
- Conf 119
- Configuration Dialog of DMC 66
- Control Processor 25
- control processor fault 27
- creating a system configuration archive 49

D

- DEBG 120

- DebugAngel 26
- DebugAngel tool 59
- Debugging 120
- detecting and isolating possible hardware faults 25
- diagnostic tasks
 - checking log file 107
 - checking protocol configuration 39
- Diagnostics Management Console 63
- Diagnostics Management Console (DMC) 21
- Digit Detector tool 69
- dlgsnapshot 27
- Dlgsnapshot tool 71
- DM3 PDK Protocol Trace 93
- DM3post 25
- DM3Trace tool 79
- DMC 63
- DMC Configuration Dialog 66
- document tag 125
- DSP test 156

E

- E1 loopback test 159
- EEPROM test 158
- EINF 120
- Enable Global Tracing 117
- ERR1 120
- ERR2 120
- error messages 152
- Errors 120
- EXCE 120

F

- FAX 119
- Filtering tab - RTF configuration 118
- firmware tracing 26

G

- GClient tag 131
- GClientLabel 131
- General tab (RTF configuration) 116

GLabel tag 130
Global Call 15, 97
Global tag 129

I

INFO 120
initialization log file 30
initialization screen 30
Intel Telecom Subsystem Summary Tool 48, 49
INTF 120
IPT 119
ISDN D-channel 83
ISDN-related messages 53
ISDNtrace tool 83
its_sysinfo 48, 49
its_sysinfo launched by Application Monitor 51

K

KernelVer tool 91

L

label 124
Label Width 118
LabelWidth 127
local diagnostic applications 65
local RAM test 156
log file
 initialization 30
 UDD 34
log file filtering (RTFManager) 120
log file panel of DMC 65
log files
 pstndiag tool 107
Log Format 117
Logfile Directory 117
Logfile Prefix 117
Logfile Size 117
Logfile tag 127
logformat 126
LS frontend test 159

M

main screen 30, 31
main window of Diagnostics Management Console

(DMC) 64
main window of RTFManager 114
Maximum Backups 117
MClient 132
MClient tag 134
MClientLabel tag 134
Media 119
menu bar of DMC 65
menu options 154
 deselect all 154, 155
 exit UDD 154
 run tests 154
 select all 154, 155
 specify tests 154
 view log file 154
MLabel 132
MLabel tag 133
module 124
Module Width 118
ModuleWidth 126
monitoring test status 33

N

network connection, checking 35
NS hardware test 160

O

OAM 119
onboard interrupt test 157
online help 151

P

path 127
PC interrupt test 158
PDK Trace tool 93
Phone tool 15
phone tool 97
preparing to start UDD 30
Program Flow 120
protocol configuration, checking 39
PSTN 119
pstndiag 39, 107
 command line options 109
 keyboard navigation 107
 lower pane 102
 upper pane 102



view bar 102
pstndiag tool 35, 39

Q

qTrace() 79

R

remote diagnostics applications 65
remote use of diagnostic tools 22
RTF Configuration - Filtering Tab 118
RTF Configuration - General Tab 116
RTF configuration file 124
RTF tool 45
RTFConfig tag 125
RtfConfigLinux.xml 123
RtfConfigWin.xml 123
RTFManager 113
run the UDD tests 32
runtime libraries 45
Runtime Trace Facility 45
Runtime Trace Facility Manager GUI 113

S

SC2000 test 157
Shared RAM (SRAM) fault 27
shared RAM test 156
shortcut keys
 Linux 151
 Windows 151
Signal Detector 69
signal processor fault 27
Signal Processors 75
size 127
specify tests 31
Springware configuration files 29
starting UDD
 Linux 30
 Windows 30
state 131
Status Monitor tool 145
stopping SpringWare boards 30
system configuration archive 49
system requirements 29

T

T1 loopback test 158
Telephony Service Provider 53
Text Aligned 117
Text Unaligned 117
Timestamp 117
Tone Generator 69
trace 126
trace entry 124
tracelocation 126
Tracing disabled 139
troubleshooting
 checklist 34
TSC 15, 97
TSP 53

U

UDD
 error messages 152
 online help 151
 preparing to start UDD 30
 supported boards 29
 system requirements 29
 user interface considerations 151
 using the UDD utility 30
 verifying board configuration data 29
UNALIGNED 126
use a diagnostics tool on a remote machine 22
user interface considerations 151
using the UDD utility 30
 aborting tests 33
 board information 153
 button options 154
 generating a log file 34
 menu options 154
 monitoring the tests 33
 run the UDD tests 32
 specify tests 31

V

view a log file 22

W

WARN 120
Warnings 120

