Intel NetStructure® Host Media Processing

Diagnostics Guide

December 2005

05-2494-001

Publication Date: December 2005

Document Number: 05-2494-001

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

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

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

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

Contents

Figures

Tables

Revision History

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

Document No.	Publication Date	Description of Revisions
05-2494-001	December 2005	Initial version of document. Much of the information contained in this document was previously published in the *Intel NetStructure Host Media Processing Diagnostics Guide*, document number 05-2356-003.

About This Publication

The following topics provide information about this publication:

Purpose

This guide provides procedures that can help you resolve problems you encounter when using Intel NetStructure® Host Media Processing (HMP) software and the Intel NetStructure® boards intended for use with the HMP software. This guide also describes the individual diagnostic tools that are provided with the HMP software.

Note: The boards are optional. HMP software can be used without boards.

Intended Audience

This publication is intended for:

- System Integrators
- Toolkit Developers
- Independent Software Vendors (ISVs)
- Original Equipment Manufacturers (OEMs)

How to Use This Publication

Refer to this document after you have installed the supported Intel NetStructure boards (optional) and the HMP software that includes the diagnostic tools.

The information in this guide is organized as follows:

- Chapter 1, “Diagnostics Overview” provides a brief overview of the diagnostic tools.

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

- Chapter 2, “Checking Intel NetStructure® Boards”
- Chapter 3, “Diagnosing First Call Issues”

- Chapter 4, “Diagnosing PSTN Protocol Issues”
- Chapter 5, “Debugging Software”

The remaining chapters provide reference information about the diagnostic tools. The chapters are organized in alphabetical order:

- Chapter 6, “CallInfo Reference” - The CallInfo tool detects call information using an Intel NetStructure board’s Telephony Service Provider (TSP) resource.
- Chapter 7, “CAS Trace Reference”- The CAS Trace tool enables you to track the bit level transitions on a robbed bit or CAS line.
- Chapter 8, “DebugAngel Reference” - The DebugAngel tool provides low level firmware tracing to aid in low level debugging.
- Chapter 9, “DigitDetector Reference” - The DigitDetector tool provides the ability to detect digits at the local end of a channel connection.
- Chapter 10, “DM3Insight Reference” - DM3Insight is a tool used to capture message and stream traffic from an Intel NetStructure board’s device driver.
- Chapter 11, “DM3post Reference”- The DM3post tool can perform diagnostics on a stopped board at any time to detect and isolate possible hardware faults.
- Chapter 12, “Getver Reference” - The Getver tool displays the version of an Intel NetStructure board’s binary file.
- Chapter 13, “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 14, “KernelVer Reference” - The KernelVer tool can be used to verify whether or not a processor has crashed.
- Chapter 15, “MercMon Reference” - MercMon provides counter information about Intel NetStructure board device drivers (Class Driver and Protocol Drivers).
- Chapter 16, “PDK Trace Reference”- The PDK Trace tool allows those who use a DM3 PDK protocol to log specific information related to the operation of the protocol.
- Chapter 17, “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 18, “PSTN Diagnostics Tool Reference” - The PSTN Diagnostics tool (pstndiag) is a utility for diagnosing and troubleshooting public switched telephone network (PSTN) connectivity problems on Intel NetStructure boards.
- Chapter 19, “QScript Reference” - This chapter provides information about QScript utilities, which are a subset of the diagnostic utilities.
- Chapter 20, “Runtime Trace Facility (RTF) Reference”- The RTF tool provides a mechanism for tracing the execution path of the runtime libraries for the HMP software. The resulting log file/debug stream output helps troubleshoot runtime issues for applications that are built with HMP software.
- Chapter 21, “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 22, "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, such as HMP and boards supported by HMP.

Related Information

Refer to the following documents and websites for more information:

- *Intel NetStructure® Host Media Processing Software Installation Guide* - This document describes how to install the HMP Software.
- *Intel NetStructure® Host Media Processing Software Administration Guide* - This publication describes how to perform the various tasks related to obtaining, activating, and otherwise working with HMP Software license as well as other administration tasks.
- *Intel NetStructure® Host Media Processing Software Release Guide* - This document provides information about the release such as product features, system requirements, and user documentation.
- *Intel NetStructure® Host Media Processing Software Release Update* - This document addresses release issues such as known problems and documentation updates.
- For technical support, go to *http://developer.intel.com/design/telecom/support/*.
- For information about Intel NetStructure Host Media Processing products, go to *http://www.intel.com/design/network/products/telecom/software/index.htm#hmp*
- For information about Intel® telecom products, go to *http://www.intel.com/design/network/products/telecom/index.htm*.

Diagnostics Overview 1

This chapter presents an overview of the diagnostic tools and the debugging/troubleshooting tasks that can be performed with them. To help you locate the tool and procedure you need, the tools are listed in the following two ways:

1.1 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 1.2, “Tool Classification by Problem Type”, on page 14.

Hardware and Firmware Diagnostics

The following tasks can be accomplished using the diagnostic tools:

Checking the hardware

You can use the Dm3post and KernelVer tools to troubleshoot the physical boards in your system. Refer to the following chapters:

- Chapter 2, “Checking Intel NetStructure® Boards”
- Chapter 11, “DM3post Reference”
- Chapter 14, “KernelVer Reference”

Checking the firmware

You can use the DebugAngel, MercMon, and PDK Trace tools to diagnose and trace problems with a board’s firmware. Refer to the following:

- Section 2.4, “Tracing Firmware”, on page 18
- Chapter 8, “DebugAngel Reference”
- Chapter 15, “MercMon Reference”
- Chapter 16, “PDK Trace 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 3, “Diagnosing First Call Issues”
- Chapter 18, “PSTN Diagnostics Tool Reference”
- Chapter 6, “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 4, "Diagnosing PSTN Protocol Issues"
- Chapter 18, "PSTN Diagnostics Tool Reference"

Collecting system data to diagnose an application failure or crash

You can 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. Refer to the following:

- Section 5.2, "Collecting System Data to Diagnose an Application Failure or Crash", on page 30
- Chapter 22, "Intel Telecom Subsystem Summary Tool Reference"

Creating a system configuration archive

As part of a quality control effort, you might want to baseline your systems by retrieving all available information using the Intel Telecom Subsystem Summary Tool (its_sysinfo). Refer to the following:

- Section 5.3, "Creating a System Configuration Archive", on page 31
- Chapter 22, "Intel Telecom Subsystem Summary Tool Reference"

Tracing Runtime Libraries

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

- Section 5.1, "Troubleshooting Runtime Issues", on page 29
- Chapter 20, "Runtime Trace Facility (RTF) Reference"

1.2 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:

- Section 2.2, "Checking an Individual Board", on page 17
- Chapter 11, "DM3post Reference"
- Chapter 14, "KernelVer Reference"

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

- Section 2.4, "Tracing Firmware", on page 18
- Chapter 8, "DebugAngel Reference"
- Chapter 15, "MercMon 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 3, "Diagnosing First Call Issues"

- Chapter 18, “PSTN Diagnostics Tool Reference”
- Chapter 6, “CallInfo Reference”

PSTN Protocol Debugging

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

- Chapter 3, “Diagnosing First Call Issues”
- Chapter 18, “PSTN Diagnostics Tool Reference”

Debugging Software

You can use the Runtime Trace Facility (RTF) tool and Intel Telecom Subsystem Summary Tool (its_sysinfo) for some general software debugging. Refer to the following:

- Section 5.1, “Troubleshooting Runtime Issues”, on page 29
- Section 5.2, “Collecting System Data to Diagnose an Application Failure or Crash”, on page 30
- Section 5.3, “Creating a System Configuration Archive”, on page 31
- Chapter 20, “Runtime Trace Facility (RTF) Reference”
- Chapter 22, “Intel Telecom Subsystem Summary Tool Reference”

Checking Intel NetStructure® Boards 2

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

2.1 Preparations

To find out which boards are in your system and which slots they are in, use the Configuration Manager (DCM).

Note: DCM is described in the HMP Administration Guide.

2.2 Checking an Individual Board

This section provides a procedure for using the DM3post tool to check an individual Intel NetStructure board at any time to detect and isolate possible hardware faults. For more information on this tool, refer to Chapter 11, "DM3post Reference".

To use the DM3post tool to check a 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.

2.3 Unsupported Boards

The HMP 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
```

2.4 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 8, "DebugAngel Reference".

DebugAngel polls the 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 8, "DebugAngel Reference".

Diagnosing First Call Issues 3

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

For more information about the pstndiag tool, refer to Chapter 18, "PSTN Diagnostics Tool Reference".

3.1 Preparation

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

- Section 18.2, "Guidelines", on page 68
- Section 18.3, "Preparing to Run the PSTN Diagnostics Tool", on page 68
- Section 18.4.1, "Starting the PSTN Diagnostics Tool", on page 68.

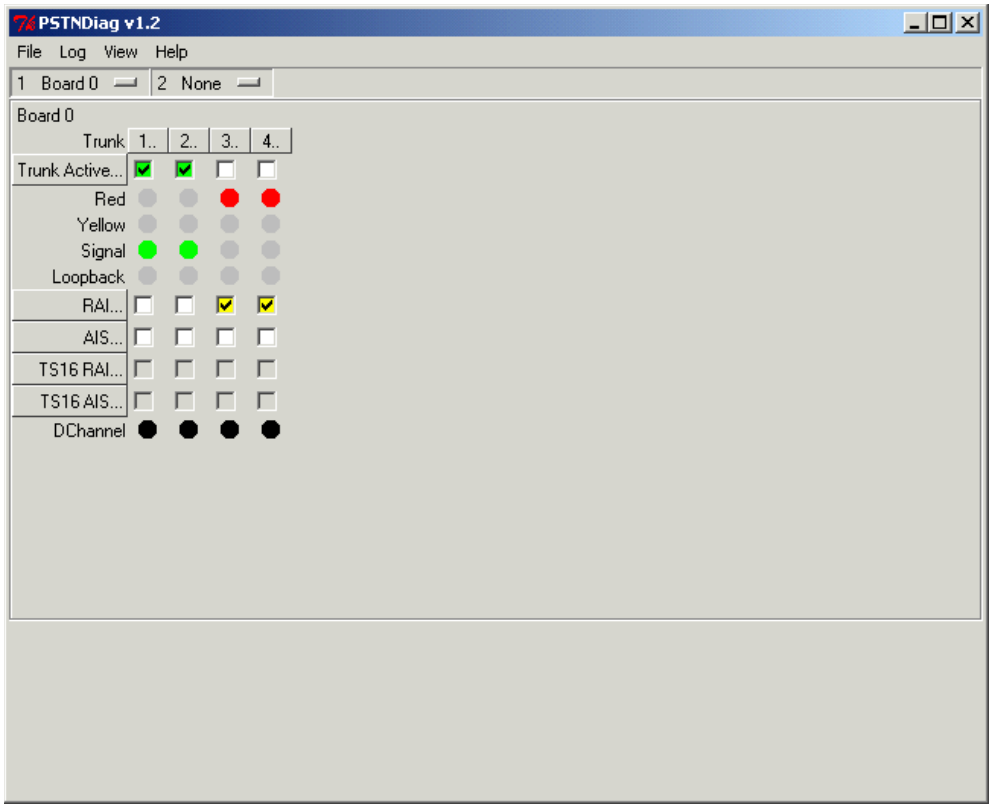
The main window displays a tree or hierarchy of the components in your system. Components include all boards that have been downloaded in the system, all trunks within a board, and all channels within a trunk.

3.2 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 1.

Figure 1. PSTN Diagnostics Tool - Board Level View



2. 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.)

3. 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 3.3, "Monitoring the Status of Alarms", on page 21.

   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.

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

3.3 Monitoring the Status of Alarms

After determining that the Signal indicator for all boards in your system is green as described in Section 3.2, "Monitoring the Signal", on page 19, 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 1, "PSTN Diagnostics Tool - Board Level View", on page 20 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.

Diagnosing PSTN Protocol Issues 4

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

For more information about the pstndiag tool, refer to Chapter 18, "PSTN Diagnostics Tool Reference".

4.1 Preparation

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

- Section 18.2, "Guidelines", on page 68
- Section 18.3, "Preparing to Run the PSTN Diagnostics Tool", on page 68

After determining that a network connection exists as described in Chapter 3, "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.

4.2 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 18.4.1, "Starting the PSTN Diagnostics Tool", on page 68.

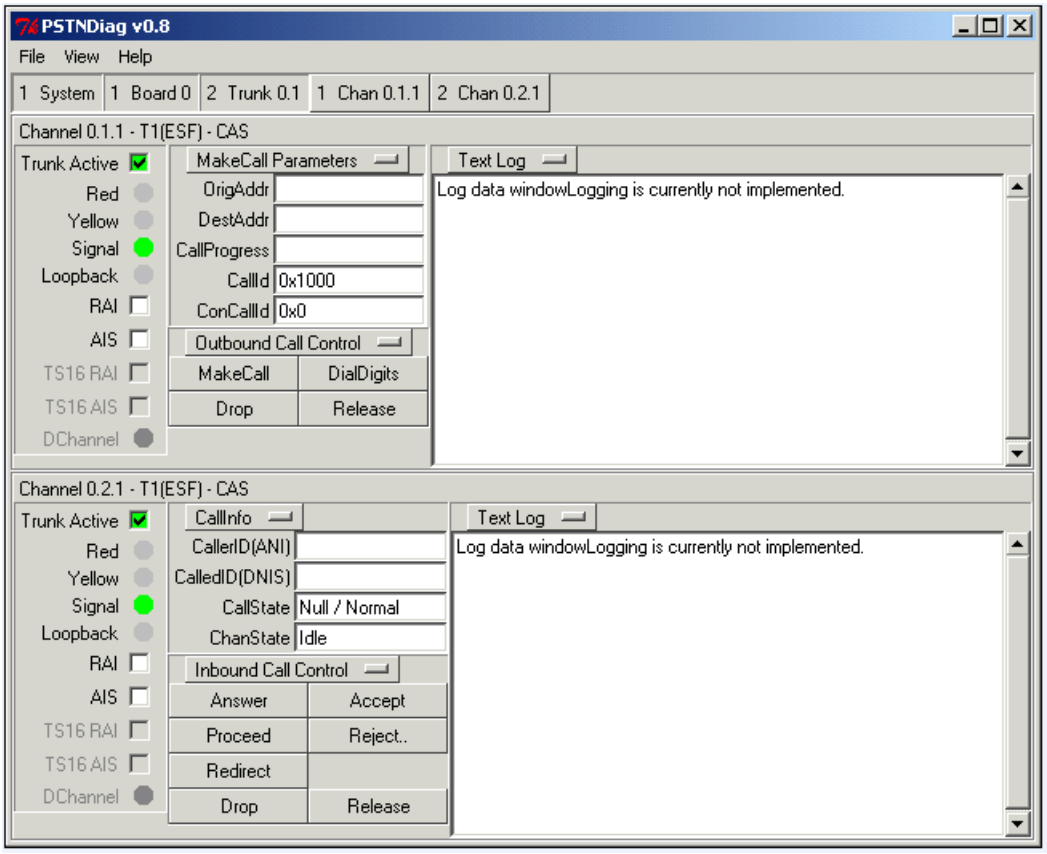
   The main window displays a tree or hierarchy of the components in your system, including all boards that have been downloaded in the system, all trunks within a board, and all channels within a trunk.

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



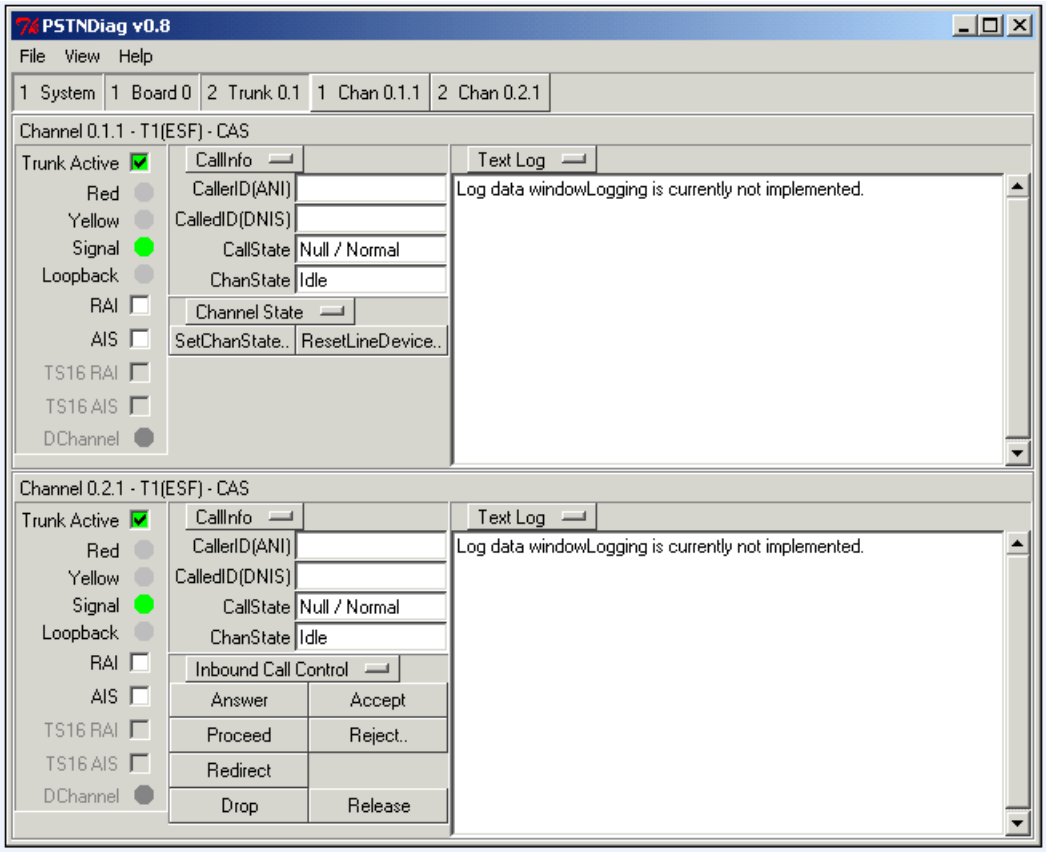
5. For the outbound channel (upper pane), click the **CallInfo** button to view additional options, and select the **Edit MakeCall Parameters** option. A new set of fields is displayed.

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 3. 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.

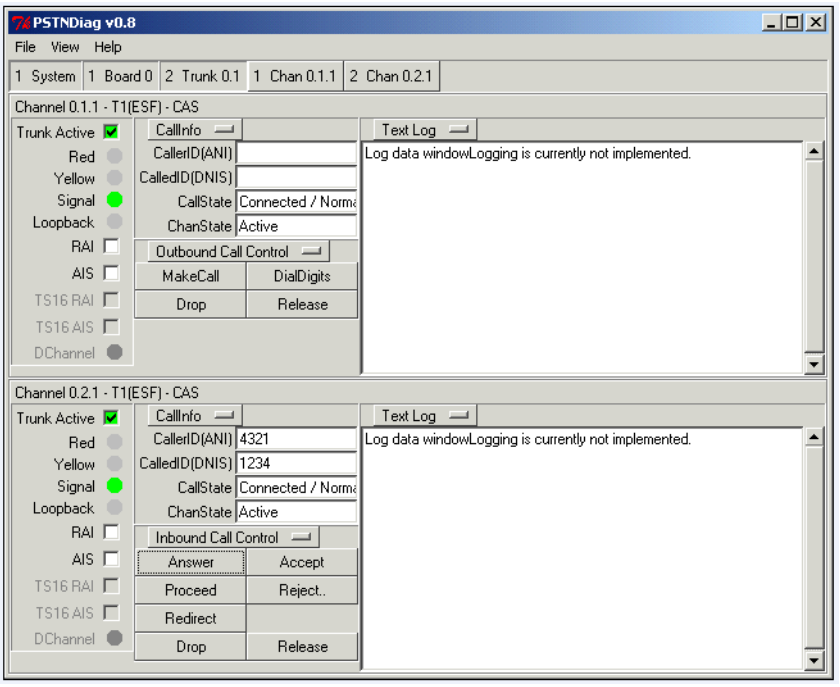
15. For the inbound channel (lower pane), click the **Answer** button to answer the incoming call. Notice that the CallState for the inbound channel now shows "Connected." If the expected state is not being displayed, it may indicate a protocol configuration problem.

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 4. PSTN Diagnostics Tool - Call Completed and Released



Debugging Software 5

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

5.1 Troubleshooting Runtime Issues

This section describes how to use the Runtime Trace Facility (RTF) for tracing the execution path of various runtime libraries. For a complete description of the RTF refer to Chapter 20, "Runtime Trace Facility (RTF) Reference".

Use the following procedure to customize and activate the RTF tool's tracing capabilities:

1. Locate the RTF configuration file. The default installation location is as follows:
   - `$(INTEL_DIALOGIC_DIR)\log` for Linux systems
   - `%INTEL_DIALOGIC_CFG%`[1] for Windows systems
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 tool's tracing capabilities. Refer to Section 20.4, "RTF Configuration File", on page 83 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 tool itself. 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 tool's tracing capabilities are turned on by default. Refer to Section 20.4.1, "RTFConfig Tag", on page 84 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 tool's logfile output. This tag is an optional part of the RTF configuration file. Refer to Section 20.4.2, "Logfile Tag", on page 86 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

1. To find out what the INTEL_DIALOGIC_CFG directory is, type `echo %INTEL_DIALOGIC_CFG%` on a command prompt and note the path displayed.

level, all activity related to that particular module is traced. Refer to Section 20.4.3, "Global Tag", on page 87 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 20.4.7, "Module Tag", on page 89 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, the RTF tool 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 tool trace levels while your application runs, it is not necessary to stop the RTF tool. 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 `RtfTrace -restart` command to restart the RTF engine.

   Note: Keep in mind that changes made to the RTF configuration file will not be reflected in the RTF tool output until the tool has been restarted.
7. At any time, you can issue the `RtfTrace` command to stop/start the RTF tool. The RTF tool will not provide output while it is stopped.

5.2 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 22, "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_gui.exe* in `%INTEL_DIALOGIC_DIR%\bin`.
     2. When program's dialog box appears, you must name the archive file into which you want the information to be collected. You can optionally specify a different archive file by clicking the file save button.
     3. Click the **Start** button to begin collecting information.
     4. A pop-up window displaying Finished Data Collection will appear to indicate completion of *its_sysinfo_gui.exe* execution. Click the **OK** button and then click the **Close** button on the *its_sysinfo_gui.exe*. 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.

5.3 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 22, "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_gui.exe* in `%INTEL_DIALOGIC_DIR%\bin`.
     2. When program's dialog box appears, you must name the archive file into which you want the information to be collected. You can optionally specify a different archive file by clicking the file save button.

3. Click the **Start** button to begin collecting information.
4. A pop-up window displaying Finished Data Collection will appear to indicate completion of *its_sysinfo_gui.exe* execution. Click the **OK** button and then click the **Close** button on the *its_sysinfo_gui.exe*. The archive file is in the location specified in the tool.

2. Retain the archive file as your baseline system information.

CallInfo Reference 6

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

6.1 Description

The CallInfo tool detects call information using the 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.

6.2 Guidelines

CallInfo is a QScript utility. For more information about QScript utilities, refer to Chapter 19, "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.

6.3 Options

The CallInfo tool uses the following command line options:

-board <n>
Logical ID of board (required).Use the Configuration Manager (DCM) to obtain the board's logical ID.

Note: DCM is described in the HMP Administration Guide.

-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
```

CAS Trace Reference 7

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

7.1 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.

7.2 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 17, “Phone Reference”.

7.3 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 Configuration Manager (DCM) to obtain the board's logical ID.

  Note: DCM is described in the HMP Administration Guide.

-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:

```
CAStrace -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)

DebugAngel Reference 8

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

8.1 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.

8.2 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.

8.3 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.

8.4 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).

DigitDetector Reference 9

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

9.1 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.

9.2 Guidelines

DigitDetector is a QScript utility. For more information about QScript utilities, refer to Chapter 19, "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 17, "Phone Reference".

9.3 Options

The DigitDetector tool uses the following command line options:

-board <n>
: Logical ID of board (required).Use the Configuration Manager (DCM) to obtain the board's logical ID.

  Note: DCM is described in the HMP Administration Guide.

-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
```

DM3Insight Reference 10

This chapter provides information about the DM3Insight tool.

DM3Insight is a tool used to capture message and stream traffic from the DM3 device driver. DM3Insight can be used for the following:

- Debugging/understanding applications, driver, kernel by capturing traffic and analyzing the trace output
- Viewing messages that are encountered only between the device driver and the DM3 board
- Eliminating orphan messages and streams
- Calculating the turnaround time for messages
- Viewing messages from the board that have incorrect source/destination address or transaction Id and therefore never reach the application
- Viewing error messages from the board like Std_MsgError or QMsgUndelivered, which are not handled (or reported) by simple applications

For complete information about configuring and using the DM3Insight tool, refer to the DM3Insight online help file (*DM3Insight.chm*). The default directory for this file is %INTEL_DIALOGIC_DIR%\bin[1] for Windows* systems.

1. To find out what the INTEL_DIALOGIC_DIR directory is, type `echo %INTEL_DIALOGIC_DIR%` on a command prompt and note the path displayed.

DM3post Reference 11

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

A procedure for using this tool is provided in Section 2.2, “Checking an Individual Board”, on page 17.

11.1 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.

Notes: ***1.*** This tool can be run on Intel NetStructure® DNI boards.

2. This tool cannot be run on the HMP “virtual board” (in the display on the Configuration Manager [DCM] Main Window, the HMP product appears as “HMP_Software” under the DM3 board group).

11.2 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. 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 HMP Administration Guide.

11.3 Options

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

-s<n>
slot number (required). Use the Configuration Manager (DCM) to obtain the board's slot number.

Note: DCM is described in the HMP Administration Guide.

-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
```

Getver Reference 12

This chapter provides reference information about the Getver tool.

The Getver tool displays the version of any Intel NetStructure® board binary file. It scans the binary for the standard DM3 version string and prints it to the screen.

Getver can also be used to get the version for most of the Linux binaries. The OA&M executable binaries and libraries have a version that getver can display.

The following example prints the version string of the *ssp.mlm* file to the screen:

```
getver ssp.mlm
```

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).

ISDN Trace Reference 13

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

13.1 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.

13.2 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.

13.3 Options

Options

The ISDNtrace tool uses the following command line options:

-b<n>
: Logical ID of board (required).Use the Configuration Manager (DCM) to obtain the board's logical ID.

  Note: DCM is described in the HMP Administration Guide.

-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
```

KernelVer Reference 14

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

14.1 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.

14.2 Options

The kernelver tool uses the following command line options:

-b<n>
: Logical ID of board (required).Use the Configuration Manager (DCM) to obtain the board's logical ID.

  Note: DCM is described in the HMP Administration Guide.

--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
```

MercMon Reference 15

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

15.1 Description

MercMon provides counter information about DM3 board device drivers (Class Driver and Protocol Drivers). These drivers maintain various counters that can aid in monitoring system activities and interpreting system behaviors.

15.2 Guidelines

The MercMon tool provides counter information about a DM3 board's class driver and protocol driver. The following sections list the types of counters for each driver:

- Class Driver Counters
- Protocol Driver Counters

15.2.1 Class Driver Counters

This section provides details about the class driver counters that are used by the MercMon tool. The class driver counters are as follows:

Note: All counters are on a per board basis except for FailMpathFind and FailStrmFind.

CompleteReads
: returns the size (in bytes) of all the read requests completed by the protocol driver (i.e., the actual size in KB of all the data read from the board)

CompleteSends
: returns the size (in bytes) of all the messages sent to or received from the board

CompleteWrites
: returns the size (in bytes) of all the write requests completed by the Protocol Driver (i.e., the actual size in KB of all the data written to the board)

FailMpathFind
: returns the number of times a request to get a message handle (Mpath) failed (global counter)

FailStrmFind
: returns the number of times a request to get a stream handle failed (global counter)

NumCanTakes
returns the number of “can takes” received for that board

NumReads
returns the total number of read requests sent down to the protocol driver

NumSends
returns the total number of messages sent down to the protocol driver

NumWrites
returns the total number of write requests sent down to the protocol driver

NumWriteSplit
returns the number of write requests, which were split for that board

NumOpenedStreams
returns the number of open streams on that board at any given time

NumCloseStreamErr
returns the number of stream close requests failed on that particular board

NumOpenStreamErr
returns the number of stream open requests failed on that particular board

NumStreamClose
returns the number of stream close requests on that particular board

NumStreamOpen
returns the number of stream open requests on that particular board

SizeReads
returns the size (in bytes) of all the messages posted down to the protocol driver

SizeSends
returns the size (in bytes) of all the messages sent down to the protocol driver

SizeWrites
returns the size (in bytes) of all the write requests sent down to the protocol driver

TimeoutReads
returns the number of read requests that timed out

TimeoutSends
returns the number of messages that timed out while waiting to be sent to the board or awaiting a reply from the board

TimeoutWrites
returns the number of write requests that timed out

15.2.2 Protocol Driver Counters

This section provides details about the protocol driver counters that are used by the MercMon tool. The protocol driver counters are as follows:

MsgsInPerSramSession
returns the number of messages read from the SRAM in one session

MsgsOutPerSramSession
: returns the number of messages written to the SRAM in one session

NoInDataDPCisr
: returns the number of times when we received an interrupt but there was no data transfer between the SRAM and the HOST

StrmsOutPerSramSession
: returns the number of streams written to the SRAM in one session (i.e., number of data blocks written)

StrmsInPerSramSession
: returns the number of streams read from the SRAM in one session (i.e., the number of data blocks read)

TotalAsyncMsgQDone
: returns the number of requests completed in the AsyncMsgQ

TotalBadSramAddr
: returns the number of times we tried reading a message/data (streams) from the SRAM but the SRAM address was wrong

TotalBadSramAddrAlloc
: returns the number of times the system tried to allocate a data block but the address was wrong

TotalBadSramAddrRelease
: returns the number of times the system tried to release a data block but the address was wrong

TotalBadSramDataCount
: returns the total number of times we got a data block of size greater than the SRAM_DATA_BLOCK_SIZE

TotalBadSramOffset
: returns the number of times we tried reading data (streams) from the SRAM but the offset calculation was incorrect

TotalBadSramOffsetAlloc
: returns the number of times we allocated data block but the offset was incorrect

TotalBigMessagesRcvd
: returns the number of times we read a “BIG” message from the SRAM (i.e., message size greater than 24 bytes)

TotalBigMessagesSent
: returns the number of times we wrote a “BIG” message to the SRAM (i.e., message size greater than 24 bytes)

TotalBogusInterrupts
: Not Used

TotalDpcOverruns
: returns the total number of DPC overruns (i.e., we were not expecting an interrupt but we got one)

TotalDmaInterrupts
: returns the total number of interrupts received from the board for performing DMA

TotalFatSramBlocks
: returns the number of times the system received a “CHAINED” data block (i.e., the data blocks are linked together)

TotalIrpsCancelled
: returns the total number of cancelled requests in any queue

TotalMsgInQ
: returns the number of requests in the Message In Queue, which are awaiting a reply from a board

TotalMsgInQDone
: returns the number of requests completed from the Message In Queue (i.e., requests completed and sent to the application)

TotalMsgInSram
: returns the total number of messages read from the SRAM

TotalMsgOutQDone
: returns the number of requests completed from the Message Out Queue (these requests are sent to the board and then moved to the Message In Queue, if expecting a reply, or completed and sent to the application)

TotalMsgOutSram
: returns the total number of messages written to the SRAM

TotalMsgOverruns
: returns the total number of overruns for the orphan message queue (i.e., we had an orphan message but there was no space in the Orphan Message Queue)

TotalMsgTimeouts
: returns the total number of requests timed out while waiting to be written to the SRAM or awaiting a reply from the SRAM (i.e. either in the Message In Q, Message Out Q or the Async Message Q)

TotalOrphanMsgsv
: returns the total number of Orphan Messages (i.e., messages which were read from the SRAM but do not have any pending requests from the application)

TotalOrphanMsgVolume
: returns the size (in bytes) of the messages present in the Orphan message queue at any given time

TotalOrphanStrms
: returns the total number of Orphan Streams in the Orphan Stream Table (i.e., data which was read from the SRAM but did not have any pending Read Requests from the application)

TotalOrphanStrmMatches
: returns the total number of streams matched in the Orphan Stream table

TotalOrphanStrmVolume
: returns the size (in bytes) of the data present in the Orphan Stream table at any given time

TotalSramDataFull
: returns a count of the number of times the HOST tried to write a stream (data) to the SRAM but the SRAM was full

TotalSramGrantInterrupts
: returns the number of “HOST SRAM PENDING” interrupts

TotalSramGrantLost
: not used

TotalSramInterrupts
: returns the total number of “NORMAL” interrupts received from the board (e.g., there is something to read from the SRAM)

TotalSramMsgFull
: returns a count of the number of times the HOST tried to write a message to the SRAM but the SRAM was full

TotalStrmInQ
: returns the number of read requests pending in the Stream In Queue

TotalStrmInQDone
: returns the number of read requests completed from the Stream In Queue (i.e., read requests completed and sent to the application)

TotalStrmInSram
: returns the number of data packets read from the SRAM

TotalStrmOutQ
: returns the number of writes performed by the user on that particular board. It is also incremented whenever the application sends an End Of Stream command.

TotalStrmOutQDone
: returns the number of Write requests completed from the Stream Out Queue (i.e., Write requests completed and sent to the application)

TotalStrmOutSram
: returns the number of data packets written to the SRAM

TotalStrmOverruns
: returns the number of Overruns for the Orphan Stream Table (e.g., we had an Orphan Stream but we had exceeded the maximum amount of memory allocated for the Orphan Stream Table or we could not allocate memory.)

TotalStrmTimeouts
: returns the total number of Read or Write Requests timed out (from the Stream In Queue or the Stream Out Queue)

TotalUnknownInterrupts
: returns the total number of unknown interrupts received from the board

15.3 Options

The MercMon tool uses the following command line options:

/f<file>
: log file name (default is *mercmon.log*)

/t<n>
display timer. The time interval (in milliseconds) between each screen refresh (default is 1000).

/l<n>
logging interval. The time interval (in seconds) between each log file update (default is 600).

/w
enable/disable logging to a file (default is enabled)

/?
displays the help screen

The following example enables logging to the *mercmon.log* file every five minutes and refreshes the screen every two seconds:

```
mercmon /t 2000 /l 300
```

PDK Trace Reference 16

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

16.1 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.

Note: PDK Trace is not supported for Intel NetStructure® DM/IP boards. 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 8, "DebugAngel Reference"):

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

16.2 Guidelines

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).

16.3 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.

16.4 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
```

Phone Reference 17

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

17.1 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.

17.2 Guidelines

Phone is a QScript utility. For more information about QScript utilities, refer to Chapter 19, “QScript Reference”.

17.3 Options

The Phone tool uses the following command line options:

-b<n>
: Logical ID of board (required).Use the Configuration Manager (DCM) to obtain the board’s logical ID.

  Note: DCM is described in the HMP Administration Guide.

-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 18

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

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

- Chapter 3, “Diagnosing First Call Issues”
- Chapter 4, “Diagnosing PSTN Protocol Issues”

18.1 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 CAStrace tools to perform further diagnostic tests

18.2 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 19, "QScript Reference".

18.3 Preparing to Run the PSTN Diagnostics Tool

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

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

18.4 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

18.4.1 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 18.7, "PSTN Diagnostics Command Line Options", on page 77.

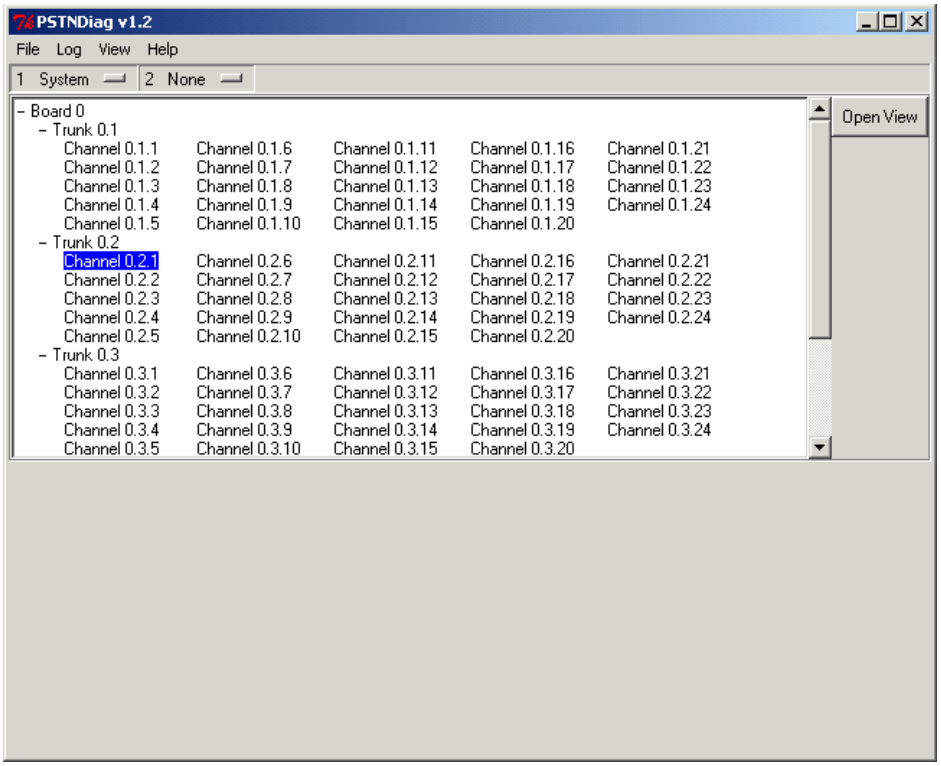
2. If a board is stopped after you have launched the pstndiag tool, you will need to exit from the tool, restart the board, and relaunch the tool to use it again.

18.4.2 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 5 illustrates a system level view of a DM/V960A-4T1-PCI board.

Figure 5. 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:

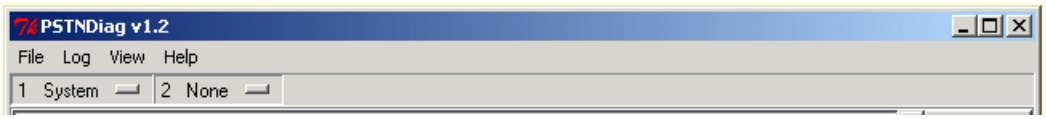
- *x* is an integer that represents the **board** logical ID, in this example **0**
- *y* is an integer that represents the **trunk** on that board, in this example **2**
- *z* is an integer that represents the **channel** number on that trunk, in this example **1**

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.

18.4.3 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 6 illustrates a sample view bar.

Figure 6. 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.

18.4.4 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 9 through Figure 16. 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.

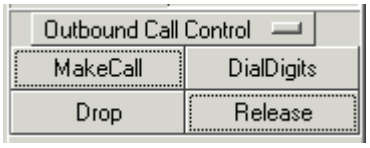
Command buttons are buttons that when pressed activate a command. There are two types of command buttons: those that toggle between two actions (these are identified by two periods following the button name) and those that issue a command directly. For example, the

ResetLineDevice.. button in Figure 7 toggles between setting a channel in service and out of service.

Figure 7. 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 8. Click on a button to issue a command.

Figure 8. 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 9. 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 10. Outbound Call Control Command Buttons

Inbound Call Control Commands

The commands under this button are as follows:

AnswerCall
Answer the call

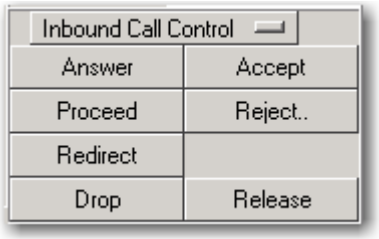
AcceptCall
Accept the call (this precedes the option of handing off the call to another client or party)

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 11. 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 12. 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:

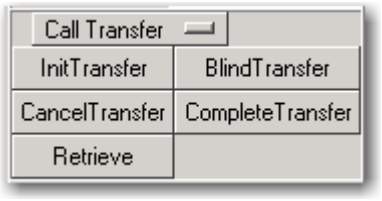
BlindTransfer (single-stage blind-transfer)
: Place the target call on hold, make a consultation call to the destination address, and then hand off the target call to the destination without determining the outcome of the consultation call. The target call's call state will transition first to HoldPendXfer, 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 13. 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 14. 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:

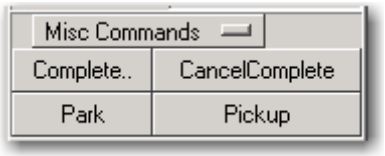
Complete
This command is not currently used.

CancelComplete
This command is not currently used.

Park
This command is not currently used.

Pickup
This command is not currently used.

Figure 15. Misc. Commands Buttons



External Applications

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

The command under this button is as follows:

Audio
Not currently used.

Figure 16. External Application Command Button

18.4.5 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.

18.5 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.

18.6 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

18.6.1 File menu

The File menu contains the following menu options:

Open log file
: Opens a previously saved log file.

Exit
: Exits the tool.

18.6.2 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 18.7, "PSTN Diagnostics Command Line Options", on page 77.

18.6.3 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.

18.6.4 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.

18.7 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<*b*>
: Starts with a specified board level view, where <*b*> represents the board logical ID.

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

-board <*b*>
: Same as -b<*b*>.

-c<*b*>.<*t*>.<*c*>
: Starts with a specified channel level view, where <*b*> 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 <*b*>.<*t*>.<*c*>
: Same as -c<*b*>.<*t*>.<*c*>.

-t<*b*>.<*t*>
Starts with a specified trunk level view, where <*b*> 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 <*b*>.<*t*>
Same as -t<*b*>.<*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`

QScript Reference 19

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

19.1 Description

The QScript utilities are a subset of the diagnostic utilities. QScript is an object-oriented scripting tool developed for the Intel NetStructure® boards. QScript is implemented using the Tcl/Tk generic scripting language. The QScript utilities require the Tcl/Tk GUI environment.

The following diagnostic utilities use QScript:

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

19.2 File Directories

The directory for the Windows batch file used to invoke the QScript tools is:

```
%systemroot%\program files\dialogic\bin
```

The QScript tools developed by Intel are located in:

```
%systemroot%\program files\dialogic\qscript
```

Note: Do not run a <*toolname*>.*qs* file directly. Batch files have been created which call the QScript interpreter to run the <*toolname*>.*qs* file. To use a QScript utility, specify the utility name and parameters on the command line.

19.3 QScript Environment Variables

This section describes the QScript environment variables:

- QSCRIPT_DIR Environment Variable
- Single Session Variable

- Remote Systems

QSCRIPT_DIR Environment Variable

In a Windows environment, this variable is set during software installation.

Single Session Variable

Set the variable for a single session using the `set` command. To permanently set the environment variable for all login sessions, update the variable in the System Properties Environment tab.

Remote Systems

The remote system containing the board does not need to have QScript installed, but must be running the RemoteQHostServer application included with QScript and installed in the bin directory.

To run QScript tools against a board in a remote system, set the REMOTE_QHOST environment variable to the name of the machine that contains the board you want to access. Set REMOTE_QHOST to:

```
hostname:port
```

where hostname is the machine name or TCP/IP address, and port is optional and specified only if RemoteQHostServer was started on a special port.

Runtime Trace Facility (RTF) Reference 20

This chapter provides an overview of the Runtime Trace Facility (RTF) tool, including the tool architecture, a procedural overview for using the tool and 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 tool's command line options is also included. This chapter provides the following subsections:

A procedure for using the tool is provided in Chapter 5, "Debugging Software".

20.1 Description

The RTF tool provides a mechanism for tracing the execution path of the runtime libraries for the Intel NetStructure® Host Media Processing (HMP) software.The resulting log file/debug stream output helps troubleshoot runtime issues for applications that are built with HMP software.

The RTF tool 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 the RTF tool to be configured to meet specific needs.

Notes: ***1.*** 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.

2. When RTF logging is enabled, users should be aware that logs are stored in the *log* directory (under *%INTEL_DIALOGIC_DIR%* for Windows or *$(INTEL_DIALOGIC_DIR)* for Linux). Users should take necessary precautions to ensure that the directory never fills.

20.2 RTF Tool Architecture

This section provides an architectural overview of the RTF tool. The following subsections are included:

20.2.1 Overview

This subsection provides a brief overview of the RTF tool architecture. The top level of the RTF architecture consists of producers and consumers:

- A producer is a software component that has internal HMP software RTF APIs incorporated into its source code. This allows producers to generate trace information. The runtime libraries are examples of producers (e.g. Global Call, Device Management, Voice).
- A consumer is the internal RTF engine that receives trace information from the RTF producers, converts the trace information into a readable format and outputs the trace information to a file or debug stream.

20.2.2 Files Used by the RTF Tool

The RTF tool uses the following files, all of which are installed as part of the HMP Software:

RtfConfigLinux.xml or *RtfConfigWin.xml*
: Editable file that allows you to customize the tracing and output capabilities of the RTF tool (e.g which runtime libraries the RTF tool will trace, the trace levels, location and size of backup log files etc.)

RtfConfig.dtd
: Document tag definition file for the RTF configuration file. This file is read only; *it should not be modified*.

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

20.3 Starting and Stopping the RTF Tool

The `RtfTrace` command is used to stop/ start the RTF tool. This command is issued from the command line and relies on an environmental variable that is defined as part of the HMP software installation routine. Therefore, the `RtfTrace` command can be issued from any directory.

The `RtfTrace` command has the following stop/start variations:

`RtfTrace -start`
: Initializes and starts the RTF tool. Tracing is only initiated if/when your application has been started.

`RtfTrace -stop`
: Stops the RTF tool.

`RtfTrace -restart`
: Restarts the RTF tool.

20.4 RTF Configuration File

To make full use of the RTF tool, 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 tool output. This subsection provides some guidelines for modifying the RTF configuration file and reference information about the file's XML tags and attributes.

Notes: ***1.*** Keep in mind that the tags present in the XML file vary according to which HMP software release you are working with. For example, some tags in this section are only present in the *RtfConfigLinux.xml* file that ships with the HMP for Linux software while other tags are only present in the *RtfConfigWin.xml* that ships with the HMP for Windows software.

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

The RTF configuration file uses a number of esoteric 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).

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 the RTF tool.

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.

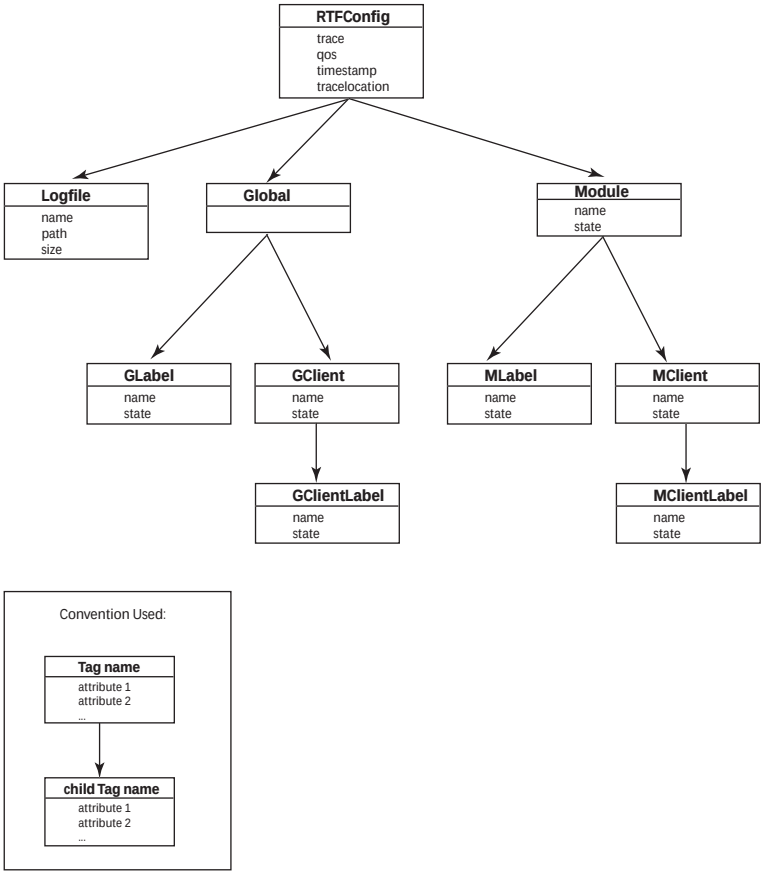
1
: when a 1 appears next to a configuration item in the RTF configuration file, it indicates that the configuration item is enabled.

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

Figure 17. RTF Configuration File Tag Structure



20.4.1 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, keep in mind that certain tags are Windows-only while other tags are Linux-only:

```
<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 determines the output mechanism for the RTF tool's trace data. The trace data can be sent to a file or a system-specific debug stream (e.g. **OutputDebugString( )** on Windows machines). Valid values are as follows:

TRACE_LOG
: RTF sends trace data to a log file. The log file details are specified in the Logfile tag. Refer to Section 20.4.2, "Logfile Tag", on page 86 for more information. This value is the default.

SYSTEM_LOG
: RTF sends trace data to a system-specific debug stream (debug console on Windows and standard output console for Linux).

timestamp

This attribute determines whether or not the output file includes a timestamp. The following two values are supported:

1
: The trace output file will include a timestamp. This is the default value.

0
: The trace output file will not include a timestamp.

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 tags 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.

– **ClientWidth number** - Allows you to customize the number of characters that appear in the Client column. The default setting is 15.
– **LabelWidth number** - Allows you to customize the number of characters that appear in the Label column. The default setting is 10.

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.

20.4.2 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="1000" maxbackups="2"/>
```

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

```
<Logfile path="$(INTEL_DIALOGIC_DIR)\log" size="1000" maxbackups="2"/>
```

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 HMP software installation routine.

For Windows systems, log files use an *rtflog-[date]-[time].txt* naming convention and cannot be changed. For Linux systems, the default log file name is *rtflog.txt*.

size

Sets the maximum size, in Kilobytes (KB), of the log file. The default setting is 1000. When the file reaches its maximum size, the RTF tool "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.

Note: Due to the internal buffers used by the RTF tool, 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.

maxbackups

Indicates the maximum number of backup log files the RTF creates. 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 file reaches the threshold defined in the Logfile tag's size attribute, the RTF trace data "rolls over" into a second log file. This sequence occurs until the number of backup log files created equals the maxbackups attribute setting.The default value is 2.

20.4.3 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, global configuration settings can be overridden by individual settings at the Module tag level (see Section 20.4.7, "Module Tag", on page 89). 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.

20.4.4 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 may 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. 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 HMP Software 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.

20.4.5 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 the 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. Example clients include:

- "dxxxB1C1"
- "dxxxB2C2"

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.

20.4.6 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. 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 HMP Software 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.

20.4.7 Module Tag

This tag is used to specify configuration for various modules. Configuration at the module level overrides global configuration. 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 HMP Software runtime libraries. Table 1 lists the runtime libraries that have modules included in the default RTF configuration file. 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.

Table 1. RTF Configuration File Traceable Modules

Library Description	File Name (.so for Linux, .dll for Windows)
PMAC Transport	pmac_transport
Fax library	libfax
Fax NTF library	libFaxntfmt (Windows only)
Voice library	libdxxm
Voice NTF library	voxspan (Windows only)
Standard Runtime library (SRL)	libsrl
NDI	ndi (Windows only)
IPM PMAC	libipm_pmac
Cheetah	Cheetah
BRI NTF	brintf (Windows only)
DTI NTF	libDtintfmt (Windows only)
DTI library	libdti
MSI Rev4 library	libmsir4 (Windows only)
MSI NTF	libMsintfmt (Windows only)
MSI DM3	MSI DM3 library
ODI	Dm3Odi
OTI	DM3Oti
DM3 Utility	DM3Utility
ipvsc library	libipm_ipvsc
ipm library	libipm
Global Call	libgc
Global Call (IP)	libgcipm
Global Call (PDK)	libpdkrt
Global Call Springware ISDN Translation layer	libgcis
ISDN library	libisdn
ISDN Technology Formatter library	isdnspan (Windows only)
UTI	UTI

Table 1. RTF Configuration File Traceable Modules

Library Description	File Name (.so for Linux, .dll for Windows)
CNF library	libcnf
Device management library	libdevmgmt
DM3 DCB	libDm3Dcb
IP CCLIB GC_H3R	gc_h3r
IP CCLIB SIP STACK	sip_stack
IP CCLIB H323 STACK	h323_stack
OAM	OAMSYSLOG

The Module tag includes the following attributes:

name

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

- “cheetah”
- “libdcnf”
- “libdevmgmt”

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.

20.4.8 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 HMP Software runtime libraries have module label names in the default RTF configuration file. Example module label names in the RTF configuration file include:

- "Error"
- "Warning"
- "Entry"

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 HMP Software runtime library module labels that exist in the default RTF configuration file have their state attribute initially set to 0.

20.4.9 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 HMP Software runtime libraries have pre-defined module client names in the default RTF configuration file. Example clients include:

- "dxxxB1C1"
- "dxxxB2C2"

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. However, the HMP Software runtime library module clients that exist in the default RTF configuration file have their state attribute initially set to 0.

20.4.10 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:

- "Error"
- "Warning"
- "Entry"

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.

20.4.11 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 three sections. The sequence of the sections must always be as follows:
   a. Logfile configuration (not required)
   b. Global configuration
   c. Module configuration
3. The Logfile section, which provides logistical information about the resulting log file(s), is optional. If you do not provide information about the logfile, then the RTF tool will use the following log file settings:
   a. File name: *rtflog-[date]-[time].txt*

b. File location: *$(INTEL_DIALOGIC_DIR)/log* for Linux, *%INTEL_DIALOGIC_DIR%\log* for Windows

c. Maximum size: 1000 KB

4. The Global configuration section can only appear one time in the RTF configuration file.
5. If a 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 *libdxxx.so* (Linux) or *libdxxx.dll* (Windows), 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 HMP Software 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. 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 logging mechanism.

20.5 Example RTF Configuration Files

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

20.5.1 Example 1: Tracing disabled - RtfConfigWin.xml

```
<?xml version="1.0" standalone="no"?>
<!DOCTYPE RTFConfig SYSTEM "RTFConfig.dtd" >


<RTFConfig trace="0" tracelocation=”TRACE_LOG” logformat=”ALIGN”>


<Logfile path="$(INTEL_DIALOGIC_DIR)\log" size="1000" maxbackups=”2”/>
```

```
<Global>

</Global>


<Module name="libdxxmt.dll" state = "1">
    MLabel name="Error" state = "1"/>
    MLabel  name="Warning" 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.

20.5.2 Example 2: Tracing enabled, logfile path and size specified, one module configured - RTFConfigLinux.xml

```
<?xml version="1.0" standalone="no"?>
<!DOCTYPE RTFConfig SYSTEM "RTFConfig.dtd"  >

<RTFConfig trace="1">

<Logfile path="$(INTEL_DIALOGIC_DIR)/log" size="1024" maxbackups=”2”/>

<Global>

</Global>

<Module name="libdxxx.so" state = "1">
   MLabel name="Error" state = "1"/>
   MLabel  name="Warning" 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* and the maximum logfile size is 1024KB. The system maintains a maximum of 2 backup log files. The log file name is not specified so the default name (*rtflog.txt*) is used.

The only module configured for trace is *libdxxx.so*. This means that all clients (devices) for this module are configured to trace Error and Warning labels.

20.5.3 Example 3: Tracing enabled, logfile path and size specified, several modules configured, global configuration used - RTFConfigWin.xml

```
<?xml version="1.0" standalone="no"?>
<!DOCTYPE RTFConfig SYSTEM "RTFConfig.dtd" >
<RTFConfig trace="1" tracelocation=”TRACE_LOG” logformat=”ALIGN”>

<Logfile path="$(INTEL_DIALOGIC_DIR)\log" size="1000" maxbackups=”2”/>

<Global>
   <GLabel name="Entry" state = "1"/>
   <GClient name="dxxxB1C2" >
      <GClientLabel name="Exit" state =”1”/>
   </GClient>
</Global>



<Module name="libdxxmt.dll" state = "1">
   MLabel name="Error" state = "1"/>
   MLabel  name="Warning" state = "0"/>
</Module>

<Module name="libdxxmt.dll">
   <MLabel name="Entry" state = "0"/>
   <MLabel  name="Warning" state = "1"/>

   <MClient name="dxxxB1C2" >
      <MClientLabel name="Entry"/>
   /MClient>
</Module>


<Module name="libFaxntfmt.dll" state = "1">
   <MLabel  name="Warning" state = "1"/
</Module>



<Module name="libdtintfmt.dll">
   <MClient name="dxxxB2C1">
      <MClientLabel name="Internal_Exit"/
   </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*. The log file's maximum size is 1000 KB. The system maintains a maximum of 2 backup log files.

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 *libdxxmt.dll* so the later configuration will take precedence. Since the state of *libdxxmt.dll* is not specified it takes default value "1". Tracing is enabled for the module *libdxxmt.dll* but only for the "Warning" label. Even though "Entry" is configured to trace in the global section, it is disabled in the *libdxxmt.dll* module configuration, so the "Entry" label will not be traced.

The client "dxxxB1C2" is also configured to trace in the *libdxxmt.dll* module with the label "Entry". Therefore the client "dxxxB1C2" in *libdxxmt.dll* is traced for the label "Entry", even though it is disabled in the module section. This results in the client "dxxxB1C2" being traced for the labels "Exit" (from global configuration), "Warning" (from module configuration) and "Entry" (from its own configuration) in module *libdxxmt.dll*. While all other clients of *libdxxmt.dll* are configured to be traced for only label "Warning", as "Warning" is the only label configured to be traced for the module.

The *libFaxntfmt.dll* module is configured to trace "Warning" (from the module section) and "Entry" (from the global section). The *libFaxntfmt.dll* client "dxxxB1C2" is configured to trace for "Warning" (from module configuration), "Exit" (from global client section) and "Entry" (from global label section). All other clients of this module are configured to trace "Warning" (from module section) and "Entry" (from the global section) since these labels are configured for the module.

The *libdtintfmt.dll* module is configured to trace "Entry" (from the global section). "dxxxB1C2" in *libdtintfmt.dll* 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 "Internalities" (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.

Status Monitor Reference 21

This chapter provides reference information about the Status Monitor tool. Topics include:

21.1 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.

21.2 Guidelines

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 Phone tool to set the channel in-service before using the bit monitoring feature. For information about the Phone tool, refer to Chapter 17, “Phone Reference”.

The Status Monitor tool is best viewed in resolutions greater than 1024 x 768.

21.3 Options

This section describes the one option for the Status Monitor tool and a sample of Status Monitor output.

Status Monitor has the following option:

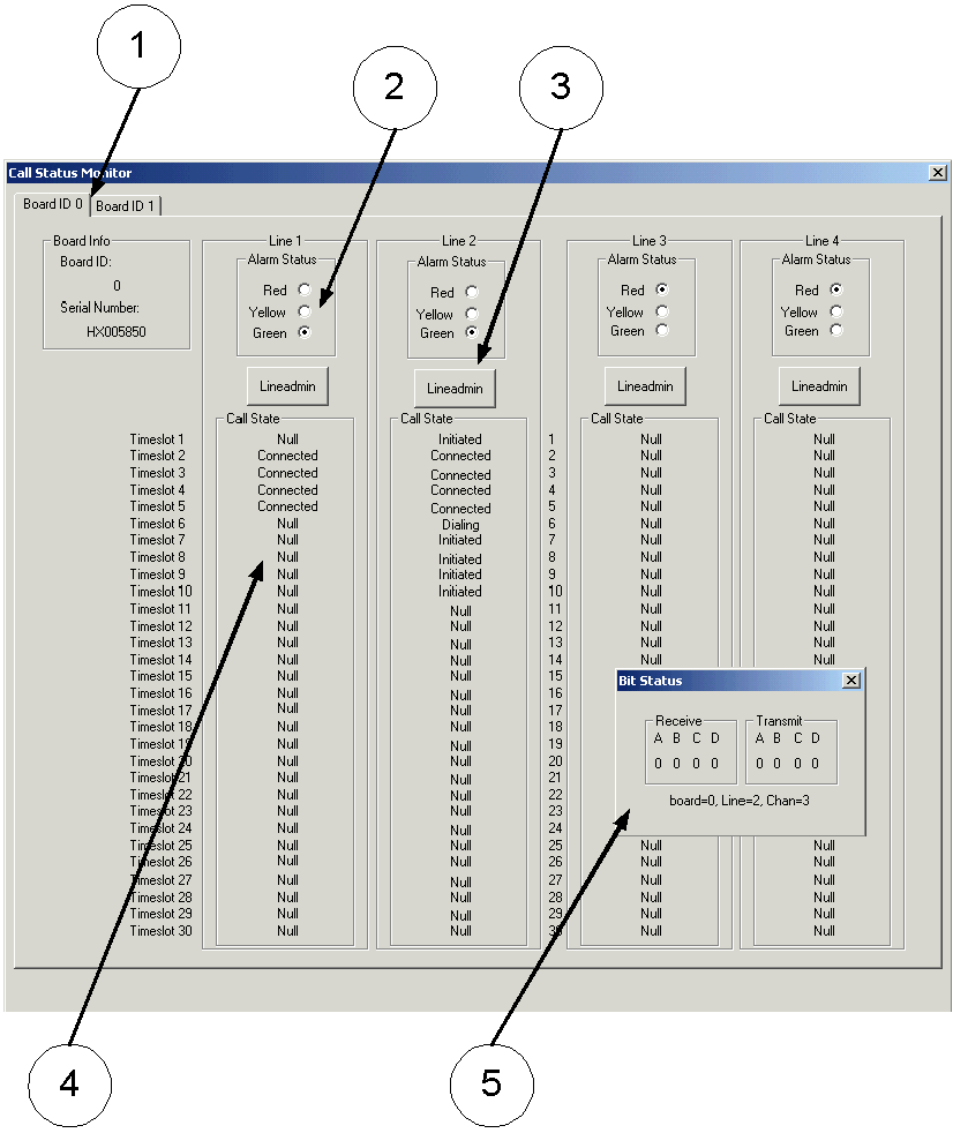
-board <board list>
: Logical ID of the board(s) to trace (optional). The default is to monitor all Intel NetStructure® DNI boards in the system. Use the Configuration Manager (DCM) utility to obtain the board’s logical ID.

The following will run the Status Monitor tool on boards 0 and 1:

```
StatusMon -board 0 1
```

Figure 18 provides an example of output from the Status Monitor tool.

Figure 18. Example of Status Monitor Output



The following numbered list corresponds to the labels in Figure 18.

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.)

Intel Telecom Subsystem Summary Tool Reference

22

This chapter describes the Intel® Telecom Subsystem Summary Tool (its_sysinfo) and the information it collects. Procedures for using this tool can be found in the following sections:

- Section 5.2, "Collecting System Data to Diagnose an Application Failure or Crash", on page 30
- Section 5.3, "Creating a System Configuration Archive", on page 31

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, HMP 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.

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

- **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.
- **Environment Variables** – its_sysinfo collects information about the operating system environment variables.
- **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 administration guide for the HMP 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 administration guide for the HMP 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 administration guide for the HMP 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.
- **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.
- **Memory and Processor** – its_sysinfo collects 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** – its_sysinfo collects information about the operating system's version, service pack, and language.

The its_sysinfo tool also checks for the log files listed in Table 2 and adds them to the archive.

Table 2. Log Files Archived by its_sysinfo

Log Files	Windows	Linux
OA& M Files	%INTEL_DIALOGIC_DIR%\log\oam.log %INTEL_DIALOGIC_DIR%\log\ncm.ini %INTEL_DIALOGIC_DIR%\log\ClusterPkg.log %INTEL_DIALOGIC_DIR%\log\ClusterPkg.log.0 %INTEL_DIALOGIC_DIR%\log\GenLoad.log %INTEL_DIALOGIC_DIR%\log\Sctsassi.log	$(INTEL_DIALOGIC_DIR)/log/oam.log $(INTEL_DIALOGIC_DIR)/log/clusterpkg.log $(INTEL_DIALOGIC_DIR)/log/clusterpkg.log.* $(INTEL_DIALOGIC_DIR)/log/board*.log $(INTEL_DIALOGIC_DIR)/log/dlgsyslogger.log $(INTEL_DIALOGIC_DIR)/log/genload.log $(INTEL_DIALOGIC_DIR)/log/iptconf.log
RTF log Files	%INTEL_DIALOGIC_DIR%\log\rtflow*****.txt	<Logfile path>/rtflog*****.txt <Logfile path> found in $(INTEL_DIALOGIC_DIR)/cfg/RtfConfigLinux.xml
		<Logfile path>/rtflog*_p.txt <Logfile path> found in $(INTEL_DIALOGIC_DIR)/cfg/RtfConfigLinux.xml
CASTrace log Files	%INTEL_DIALOGIC_DIR%\log	$(INTEL_DIALOGIC_DIR)/log/CAStrace.log.*

Table 2. Log Files Archived by its_sysinfo (Continued)

Log Files	Windows	Linux
DebugAngel Files	%INTEL_DIALOGIC_DIR%\log\debugangel.*	$(INTEL_DIALOGIC_DIR)/log/debugangel.*
Pstndiag Trace log Files	%INTEL_DIALOGIC_DIR%\log	$(INTEL_DIALOGIC_DIR)/log/pstndiag.*
IP Protocol Files	C:\WINDOWS\System32: gc_h3r.log, rtvsp1.log, siplog.txt, and sdplog.txt	$(HOME)/g*.log $(HOME)/rtvsp1.log
Dr. Watson Dump and Log Files	C:\Documents and Settings\UserAccount\Local Settings\Application Data\Microsoft\Dr Watson\user.dmp... C:\Documents and Settings\UserAccount\Local Settings\Application Data\Microsoft\Dr Watson\DrWtsn32.log...	
Dump Files		/root/core…
Its_sysinfo logfile files…	its_sysinfo.log	its_sysinfo.log

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.

Index

N

P

R

S

T

U

V

W