Dialogic
Making Innovation Thrive™

Dialogic® Host Media Processing Software

Diagnostics Guide

October 2009

05-2356-005

Publication Date: October 2009

Document Number: 05-2356-005

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
05-2356-005	October 2009	Made global changes to reflect Dialogic brand. Changed title to "Dialogic® Host Media Processing Software Diagnostics Guide." Debugging Software chapter: Added. Includes task information for the RTF tool and its_sysinfo tool. Tracing the Runtime Libraries: Removed chapter as this information is now included in "Debugging Software" chapter. DM3post Reference chapter: Added. Listboards Reference chapter: Added. Runtime Trace Facility (RTF) Reference chapter: Updated rtftool clean command in Cleaning the RTF Log File. Reorganized some sections and made other minor edits. Telecom Subsystem Summary Tool (its_sysinfo) Reference chapter: Added.
05-2356-004	August 2005	Global: Editorial and branding changes.
05-2356-003	April 2005	RTFConfig Tab: The default setting of the trace attribute has changed from 0 to 1. Example RTF Configuration Files: Added two Windows examples.
05-2356-002	March 2005	Diagnostics Overview: References to DM3StdErr and Qerror removed. Added paragraph and figure to explain how HMP products are shown on the DCM configuration manager GUI. Tracing the Runtime Libraries: Title of this chapter changed. Replaced the path of the location for the RTF tool with an environment variable. DM3StdErr Reference: Chapter removed. Getver Reference: Changed the example command and added a Note. Qerror Reference: Chapter removed. StrmStat Reference: Chapter removed. Runtime Trace Facility (RTF) Reference: Added a Note to the size topic in the Logfile Tag section. Guidelines for Editing the RTF Configuration File: A guideline was added about potentially experiencing I/O throughput degradation when using full RTF logging on high-density systems.
05-2356-001	September 2004	Initial version of document.

About This Publication

This preface provides the following information about this publication:

Purpose

This publication describes diagnostic tools, each of which is presented in a separate reference chapter.

Applicability

This document version (05-2536-005) is published for Dialogic® Host Media Processing (HMP) Software Release 4.1LIN.

This document may also be applicable to other software releases (including service updates). Check the Release Guide for your software release to determine whether this document is supported.

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 Dialogic® boards (optional) and the Dialogic® HMP Software that includes the diagnostic tools.

This document provides an overview of the diagnotic tools, discusses diagnostic tasks, and presents reference information about each tool.

Related Information

See the following for additional information:

- *http://www.dialogic.com/manuals/* (for Dialogic® product documentation)
- *http://www.dialogic.com/support/* (for Dialogic Services and Support)
- *http://www.dialogic.com/* (for Dialogic® product information)

Diagnostics Overview

This section provides a brief description of the diagnostic tools included with the Dialogic® system software.

The diagnostic tools are described as follows:

DM3post Reference
: The DM3post tool performs diagnostics on a stopped board to detect and isolate possible hardware faults.

Getver Reference
: The getver tool displays the version of a board's binary file.

Kernelver Reference
: The kernelver tool queries the board's kernel running on a particular processor for its version number.

Listboards Reference
: The listboards tool displays information about the boards in the system.

Runtime Trace Facility (RTF) Reference
: The RTF tool provides a mechanism for tracing the execution path of various runtime libraries. The trace information can be captured in log files or sent to a debug stream.

Telecom Subsystem Summary Tool (its_sysinfo) Reference
: The Telecom Subsystem Summary tool (its_sysinfo) provides information about the system environment.

Debugging Software 2

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

2.1 Troubleshooting Runtime Issues

This section describes how to customize and activate the Runtime Trace Facility (RTF) tracing capabilities.

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

1. Locate the RTF configuration file (*RtfConfigWin.xml* for Windows® and *RtfConfigLinux.xml* for Linux). The default installation location is determined by the INTEL_DIALOGIC_CFG environment variable.

   The INTEL_DIALOGIC_CFG environment variable is set when the Dialogic® system software is installed. See the Software Installation Guide for information about Dialogic® system software environment variables. To determine the variable setting on a Linux system, type `echo $INTEL_DIALOGIC_CFG`. To determine the variable setting on a Windows® system, type `set INTEL_DIALOGIC_CFG`.

2. If you are familiar with XML syntax, use any ASCII text editor to open the RTF configuration file. If you are not familiar with XML syntax, open the RTF configuration file with XML editor software.

3. Edit the RTF configuration file to customize the RTF tracing capabilities. See Section 7.3, "RTF Configuration File", on page 30 for complete information about editing the RTF configuration file.

4. When you are finished editing the RTF configuration file, save and close the file, and issue the `rtftool reload` command to apply the changes.

5. Start your application. As your application runs, RTF will trace the runtime libraries according to RTF configuration file settings. Refer to the individual entries in the log file or debug stream to review the trace statements generated by the runtime libraries.

For more information about RTF and manually editing the configuration file, see Section 7.4, "Guidelines for Editing the RTF Configuration File", on page 42 in Chapter 7, "Runtime Trace Facility (RTF) Reference".

2.2 Collecting System Data to Diagnose an Application Failure or Crash

This section describes how to use the Telecom Subsystem Summary Tool (its_sysinfo) to collect the system data you will need to send to Dialogic Services and Support 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 Telecom Subsystem Summary Tool (its_sysinfo) to collect the log files and environment information to send to Dialogic Services and Support.
- Your telephony application doesn't work as expected. For example, an attempt to make a call fails. At that point, you would run the Telecom Subsystem Summary Tool to gather the log files and environment information to send to Dialogic Services and Support.

To familiarize yourself with the Telecom Subsystem Summary Tool (its_sysinfo) and all the data it collects, see Chapter 8, "Telecom Subsystem Summary Tool (its_sysinfo) Reference".

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

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

2.3 Creating a System Configuration Archive

This section describes how to use the 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 Telecom Subsystem Summary Tool (its_sysinfo) and all the data it collects, see Chapter 8, "Telecom Subsystem Summary Tool (its_sysinfo) Reference".

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

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

DM3post Reference 3

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

3.1 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.

This tool can be run on Dialogic® DNI boards.

This tool cannot be run on the Dialogic® HMP "virtual board". For example, in Dialogic® HMP Software for Windows®, the product appears as "HMP_Software" under the DM3 board group in the Dialogic Configuration Manager Main Window.

3.2 Guidelines

Make sure the board is in a "stopped" state before running the DM3post tool. 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 Dialogic® DM3 Boards in the chassis. By using the chassis option with the reset option, you can run POST on all Dialogic® 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.

Note: 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.

3.3 Options

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

-s<n>
slot number (required). On Windows®, use the Dialogic® Configuration Manager (DCM) to obtain the board's slot number. On Linux, use the listboards utility to obtain the board's physical slot number.

-b<n>
bus number (optional if there is only one bus *or* if the slot number is unique)

-c
chassis (optional). If this option is specified, slot number and bus number are not required.

-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 (optional)

-v
displays the program version (optional)

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: Run DM3post with the Reset Option on a Board

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

This chapter provides reference information about the getver (Get Version) tool. The following topics are discussed:

4.1 Description

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

```
getver ssp.mlm
```

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

- Dialogic® Board firmware files: **.srec*, **.mlm*
- Dialogic® Board files: **.hot*, **.psi*
- Windows® Dynamic Load Library files: **.dll*
- Linux Dynamic Load Library files: **.so*
- Other files in the Dialogic® software that follow supported versioning formats (the OA&M executable binaries and libraries have a version that getver can display)

The getver tool is located in the *bin* directory by default.

4.2 Options

This section provides the command line arguments for getver:

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

Note the following:

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

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

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

4.3 Output

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

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

Here are some examples of output:

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

Kernelver Reference

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

5.1 Description

The kernelver (Kernel Version) 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.

5.2 Options

The kernelver tool uses the following command line options:

-b<n>
: Logical ID of board (required). On Linux, use the listboards tool to obtain the board's logical ID. On Windows®, use the Dialogic® Configuration Manager (DCM) to obtain the board's logical ID.

--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
```

Listboards Reference 6

This chapter provides reference information about the listboards tool. The following topics are discussed:

6.1 Description

The listboards tool displays information about the Dialogic® boards that are installed in the system.

You must install the dmdev RPM to use the listboards tool regardless of the type of board in your system. To install the dmdev RPM, select a menu item during installation of the software release (see the appropriate *Software Installation Guide*).

The listboards tool provides two levels of information: Level 1 and Level 2. See Table 1 for a list of Level 1 parameters and Table 2 for a list of Level 2 parameters.

To use Level 1, specify `-l1` or `--release 1` parameter at the command line. Level 1 is also invoked by default (that is, when you invoke the tool without any parameters or without the Level 2 parameter).

```
listboards [-l1 | --release1] [level1_parameter_list]
```

To use Level 2, specify `-l2` or `--release 2` parameter at the command line.

```
listboards -l2  | --release2 [level2_parameter_list]
```

6.2 Guidelines

The following guidelines on using the listboards tool are provided:

- You can specify a parameter in one of two forms: abbreviated (`-x`) or verbose (`--yyyyy`). Examples: `-h` or `--help` to display help; `-v` or `--version` to find out the tool version. Some parameters are available in both forms, but others are available only in one form.
- Always precede a parameter by a dash (-) or double dash (--); otherwise the text will be ignored.
- For the verbose form, include a space between the parameter and its specified value; for example, `listboards -l2 --iAUID 50023`. For the abbreviated form, you can include the parameter with or without the space.

- Options that are described as “intrusive” send messages to the board and thus may temporarily affect performance.
- This tool displays or provides a logical ID (Log ID) number, which is synonymous with board number and is specified when running any of the utilities that require a logical ID number. The listboards tool displays the logical board number along with the serial number of the board. Board numbers are dynamically assigned on start up, so use the serial number to physically identify the board in the chassis. Also, the `--oLID` parameter is used to obtain the logical ID number of a specified board.
- This tool displays a list of all boards if no parameters are used to identify a specific board. For example, `listboards -l 2` displays a list of all boards, while `listboards -l 2 -a 50023 --oLID` returns the logical board ID for the board having AUID 50023.
- More complete information is displayed by the listboards tool if you start the boards first.

6.3 Options

Table 1 provides a list of Level 1 parameters.

Table 1. Listboards Level 1 Parameters

Parameter	Description
-b <*boardnumber*> --scbus <*boardnumber*>	Lists SCBus information for board with the specified logical ID (intrusive). **Note:** The -b option performs a different function in Level 2.
-c <*pcislotnumber*> --cfgrom <*pcislotnumber*>	Lists ROM configuration from specified PCI slot number.
-h --help	Displays help. Performs the same function in Level 1 and Level 2.
-i <*boardnumber*> --info <*boardnumber*>	Retrieves detailed hardware information for board with the specified logical ID (intrusive).
-l <*level*> --release <*level*>	Selects *listboards* release level to run: 1 (default) or 2. This table only describes Level 1 parameters.
-v --version	Lists tool version information. Performs the same function in Level 1 and Level 2.

Example for Level 1

The following command runs the listboards tool using level 1:

```
listboards
```

The following is an example of the level 1 output:

```
BrdNum CfgId  Type Bus Slot PhysAddr   RamSize   Irq  State
   1     0      P   0    0  ff000000    80000    0xa  DOWNLOADED
   2     1      P   2    2  faf00000    80000    0xb  DOWNLOADED
   3     2      P   2    6  fae80000    80000    0xb  DOWNLOADED
```

Table 2 provides a list of Level 2 parameters.

Table 2. Listboards Level 2 Parameters

Parameter	Description
-a <*auid*> --iAUID <*auid*>	Lists board with matching AUID. This parameter is also used with the --oLID or --oPHYS parameters.
-b <*pcibus*> -s <*pcislot*> --iPCIBus <*pcibus*> --iPCISlot <*pcislot*>	Lists board with matching PCI bus number and PCI slot number. Both parameters are required. These parameters can also be used with the --oLID or --oPHYS parameters. **Note:** The -b option performs a different function in Level 1, and the -s option is not supported in Level 1.
-e --extrainfo	Provides extra Dialogic® DM3 information on model number, driver configuration ID, driver state, and admin state.
-h --help	Displays help. Performs the same function in Level 1 and Level 2.
-l <*level*> --release <*level*>	Selects *listboards* release level to run: 1 (default) or 2. This table only describes Level 2 parameters.
-p <*phys_id*> | <*wheel_id*> --iPHYS <*phys_id*> | <*wheel_id*>	Lists board with matching physical ID for Dialogic® board CompactPCI form factor or thumbwheel ID for PCI.
-q --quiet	Lists in quiet mode (without the title bar).
-r --oROM	Outputs ROM of found board.
-s <*pcislotnumber*> --iPCISlot <*pcislotnumber*>	Specifies the PCI slot number of a board (for --oLID or --oPHYS). This parameter must be used with the -b <*pcibus*> parameter. These parameters can also be used with the --oLID or --oPHYS parameters. **Note:** The -b option performs a different function in Level 1, and the -s option is not supported in Level 1.
-v --version	Lists tool version information. Performs the same function in Level 1 and Level 2.
<*board*> --oLID	Outputs logical ID of specified <*board*>. The board must be specified with the -a option or with the -b and -s options (otherwise it uses the first board by default). **Note:** This parameter must be used in the sequence shown; that is, the specified board (input) must precede the --oLID parameter (output).
<*board*> --oPHYS	Outputs physical ID (Dialogic® board CompactPCI form factor with Hot Swap Kit) or thumbwheel ID (PCI, or Dialogic® board CompactPCI form factor without Hot Swap Kit) of specified <*board*>. The board must be specified with the -a option or with the -b and -s options (otherwise it uses the first board by default). **Note:** This parameter must be used in the sequence shown; that is, the specified board (input) must precede the --oPHYS parameter (output).

Example for Level 2

The following command runs the listboards tool using level 2:

```
listboards -l2
```

The following is an example of level 2 output:

```
| AUID |CT Platform|PCIBus|PCISlot|ThumbWheel|Log ID|Serial Num|ModelNum

 50002  DM3         0      13      0          1      KS014983   0x100
 50011  DM3         2      9       2          2      KS014980   0x100
 50020  DM3         2      10      6          3      GG000104   0x1E01

Total number of boards: 3
```

Runtime Trace Facility (RTF) Reference

This chapter provides an overview of the Runtime Trace Facility (RTF), including information about editing the RTF configuration file to set tracing configuration options. Reference information about the RTF command line options is also included. This chapter contains the following topics:

A procedure for using the tool is provided in Chapter 2, “Debugging Software”.

7.1 Description

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

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

Major RTF components include:

- Client library (*librtfmt.dll* for Windows®, *librtf.so* for Linux): Dynamic load library that provides the software interface to make use of RTF functionality.
- Configuration file (*RtfConfigWin.xml* for Windows®, *RtfConfigLinux.xml* for Linux): Editable file that allows you to customize the tracing and output capabilities of RTF (for example, which runtime libraries RTF will trace, the trace levels, location and size of backup log files).

  Note: You must have administrative rights to modify the *RtfConfig*.xml* files.
- Server application (*RtfServer.exe* for Windows®, *RtfServer* for Linux): Service that communicates to all RTF clients, receives trace data from clients, and dispatches data to disk.

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

7.2 Installing RTF

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

7.3 RTF Configuration File

To make full use of RTF, you will want to modify the RTF configuration file (*RtfConfigLinux.xml* for Linux, *RtfConfigWin.xml* for Windows®). This configuration file allows you to define which trace messages will be included in the RTF output.

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

This section provides reference information about the file's XML tags and attributes. For guidelines on modifying the file, see Section 7.4, "Guidelines for Editing the RTF Configuration File", on page 42.

7.3.1 Terminology

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

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

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

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

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

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

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

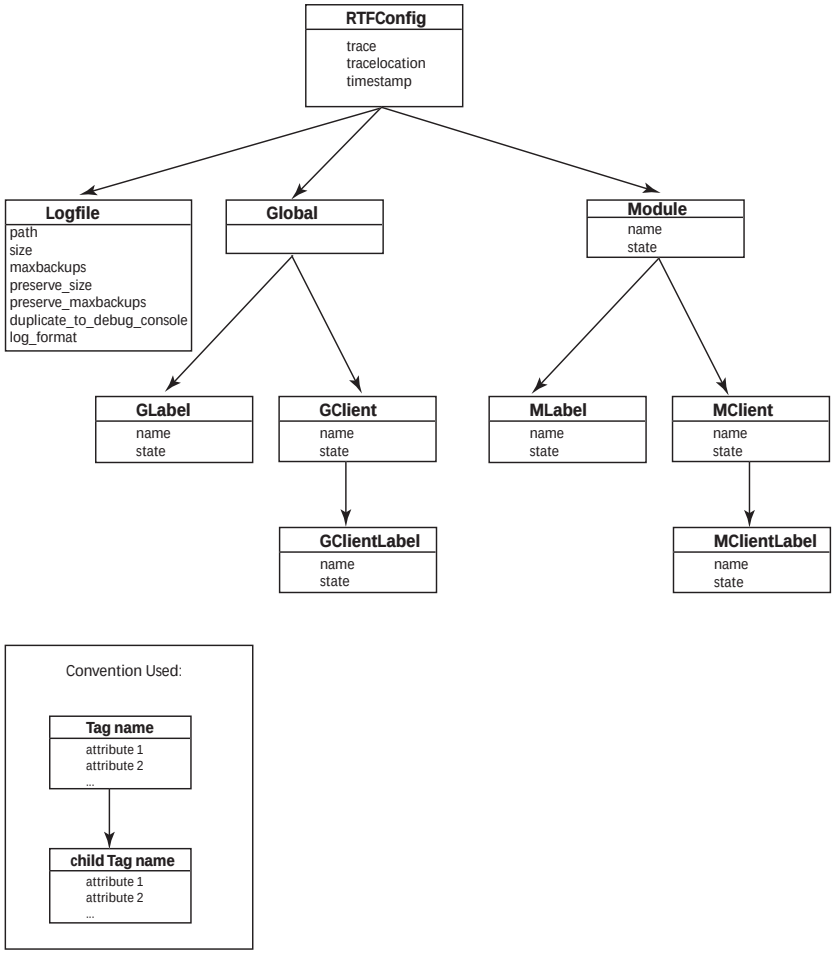
7.3.2 RTF Configuration File Tag Structure

The RTF configuration file’s top-level document tag is the RTFConfig tag. The following tags are child tags of the RTFConfig tag. Also shown are the child tags of these tags.

- Logfile Tag
- Global Tag
  - GLabel Tag
  - GClient Tag
    - GClientLabel Tag
- Module Tag
  - MLabel Tag
  - MClient Tag
    - MClientLabel Tag

Figure 1 shows the XML tag structure and tag attributes of the RTF configuration file:

Figure 1. RTF Configuration File Tag Structure



7.3.3 RTFConfig Tag

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

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

<!-- Logfile section goes here -->

<!-- Global section goes here -->

<!-- Module sections go here -->

</RTFConfig>
```

The RTFConfig tag includes the following attributes:

trace

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

0
: RTF tracing is disabled.

1
: RTF tracing is enabled. This is the default setting.

tracelocation

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

timestamp

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

7.3.4 Logfile Tag

The Logfile tag is the first child tag of the RTFConfig tag. The Logfile tag sets parameters for the RTF log file output. This tag is an optional part of the RTF configuration file.

To customize the appearance and settings of the log file, edit the Logfile tag within the RTF configuration file. The Logfile tag appears under the RTFConfig tag in the RTF configuration file.

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

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

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

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

The Logfile tag includes the following attributes:

path

Indicates a valid directory path for the log file. The default path for Linux is *$(INTEL_DIALOGIC_DIR)/log*. The default path for Windows® is *$(INTEL_DIALOGIC_DIR)\log*. The INTEL_DIALOGIC_DIR environment variable is defined as

part of the Dialogic® system software installation routine. See the Software Installation Guide for more information about the INTEL_DIALOGIC_DIR environment variable.

size

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

Due to the internal buffers used by RTF, the actual size of the log file may be up to 1 MB larger than the size attribute's value. For example, if the size attribute is set to 1000 KB, the actual log file may grow up to 2000 KB.

Note: This attribute is only applicable when RTF preservation mode is turned OFF. When RTF preservation mode is ON, the preserve_size attribute determines the size of the log file. See Section 7.6.3, "Running RTF in Preservation Mode", on page 44 for information about preservation mode.

maxbackups

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

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

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

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

Note: This attribute is only applicable when RTF preservation mode is turned OFF. When RTF preservation mode is ON, the preserve_maxbackup attribute determines the number of backup log files. See Section 7.6.3, "Running RTF in Preservation Mode", on page 44 for information about preservation mode.

preserve_size

Sets the maximum size, in Kilobytes (KB), of the log file when the RTF is run with preservation mode turned ON. The default setting is 300. When the file reaches its maximum size, RTF "rolls

over" the log file, much like a circular buffer. The previous log file is saved as a timestamped text file. RTF aligns the entries so that the oldest entry is always at the top of the log file and the newest entry is always at the bottom of the log file.

Note: This attribute is only applicable when RTF preservation mode is turned ON. When RTF preservation mode is OFF, the size attribute determines the size of the log file. See Section 7.6.3, "Running RTF in Preservation Mode", on page 44 for information about preservation mode.

preserve_maxbackups

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

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

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

The default preserve_maxbackups value is 10.

Note: This attribute is only applicable when RTF preservation mode is turned ON. When RTF preservation mode is OFF, the maxbackups attribute determines the number of backup log files. See Section 7.6.3, "Running RTF in Preservation Mode", on page 44 for information about preservation mode.

duplicate_to_debug_console

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

1
: The trace output information will be duplicated on the debug console for Windows® systems or the standard error output for Linux systems.

0
: The trace output information will not be duplicated on the debug console for Windows® systems or the standard error output for Linux systems. This is the default setting.

log_format

This attribute determines the format of the output log file. By default, this is set to "text" (the trace output log is written in readable, text file format). You can change the attribute to "binary" to get binary output. To view binary output, you must convert it to text format using the `rtftool export` command. For more information, see Section 7.6.4, "Exporting Binary Log Files to Text Format", on page 45.

7.3.5 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. When a module is traced at the global level, all activity related to that particular module is traced. However, global tag settings have the lowest priority; they can be overridden by settings in individual Module tags (see Section 7.3.9, "Module Tag", on page 38).

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.

7.3.6 GLabel Tag

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

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

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

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

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

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

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

The GLabel tag has the following attributes:

name

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

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

- "Debug"
- "Info"
- "Warning"

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

state

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

1
: Label is enabled at the global level. All trace messages associated with this label will be sent to the trace output. This is the default value.

0
: Label is disabled at the global level. Trace messages associated with this label will not be sent to the trace output.

7.3.7 GClient Tag

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

name

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

- "dxxxB1C1" (voice device)
- "dxxxB2C2" (voice device)
- "ipmB2C2" (IP media device)

Note: See the *Dialogic® Standard Runtime Library API Programming Guide* for more information about client names for Dialogic® libraries and devices.

state

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

1
: Client is enabled at the global level. All trace messages associated with this client will be sent to the trace output. This is the default value.

0
: Client is disabled at the global level. Trace messages associated with this client will not be sent to the trace output.

7.3.8 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 attributes:

name

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

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

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

state

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

1
: Client label is enabled at the global level. Trace messages associated with this label will be sent to the trace output. This is the default value.

0
: Client label is disabled at the global level. Trace messages associated with this label will not be sent to the trace output.

7.3.9 Module 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 *RtfConfigLinux.xml* and *RtfConfigWin.xml* must contain at least one Module tag.

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

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

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

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

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

The Module tag includes the following attributes:

name

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

- “gc”
- “libsrl”
- “spwrdevmgmt”

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.

7.3.10 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 attributes:

name

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

- "APPL"
- "DEBG"
- "WARN"

state

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

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

0
Label is disabled. Trace messages associated with this label will not be sent to the trace output.

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

7.3.11 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 attributes:

name

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

- "dxxxB1C1" (voice device)
- "dxxxB2C2" (voice device)
- "ipmB2C2" (IP media device)

Note: See the *Dialogic® Standard Runtime Library API Programming Guide* for more information about client names for Dialogic® libraries and devices.

state

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

1
Client is enabled. Trace messages associated with this client will be sent to the trace output. This is the default value.

0
: Client is disabled. Trace messages associated with this client will not be sent to the trace output.

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

7.3.12 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 attributes

name

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

- "APPL"
- "DEBG"
- "WARN"

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

state

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

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

0
: Label is disabled. Trace messages associated with this label will not be sent to the trace output.

7.3.13 Precedence Scheme for Module/Client Labels

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

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

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

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

- Module: disabled

- Client: enabled
- Label: disabled

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

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

Figure 1, "RTF Configuration File Tag Structure", on page 32 shows the hierarchy of the possible handles and how they relate to one another.

7.4 Guidelines for Editing the RTF Configuration File

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

- Do not change the name of the file. The filename must remain at its default setting.
- The RTF configuration file is broken down into four sections. The sequence of the sections must always be as follows:
  1. RTFconfig
  2. Logfile configuration
  3. Global configuration
  4. Module configuration
- RTF uses the following default log file names:
  - *rtflog-<year><month><day>-<hour>h<minute>m.<second>s.txt* when preservation mode is OFF
  - *rtflog-<year><month><day>-<hour>h<minute>m.<second>s_p.txt* when preservation mode is turned ON
- The Global configuration section can only appear one time in the RTF configuration file.
- If there is configuration information that 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.
- Configuration information that is repeated later in the file will take precedence. For example, if there are two module configurations for *spwrvoice*, the configuration information lower in the file will dictate tracing behavior for that module.

  Note: This should not be done, but it is mentioned here so the behavior can be recognized if this situation occurs by error.
- RTF is case sensitive. Therefore, the trace labels "Error" and "error" are considered to be two distinctly different trace labels.

- Simultaneous tracing of multiple Dialogic® libraries may have an adverse effect on system performance.
- 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.
- If you wish to dynamically edit the RTF trace levels while your application runs, it is not necessary to stop RTF. Instead, perform the following:
  1. Open the RTF configuration file.
  2. Customize the settings in the RTF configuration file.
  3. Save and close the RTF configuration file.
  4. Issue the `rtftool reload` command to reload the RTF configuration file and restart the RTF engine.

  Note: Keep in mind that changes made to the RTF configuration file will not be reflected in the RTF output until the tool has been reloaded.
- At any time, you can issue the `rtftool` command to pause, reload, and resume RTF. RTF will not provide trace output while it is paused. See Section 7.6, "rtftool Command", on page 44 for more information.
- Save the original *RtfConfigLinux.xml* file or *RtfConfigWin.xml* file, as it is advisable to return to the original logging level when finished using the RTF logging mechanism.

7.5 Restrictions and Limitations

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

- If you run full RTF logging on high-density systems, you may experience I/O throughput degradation. It is recommended that you do not run RTF with full logging on high-density systems or any field-deployed systems. Or you can consider specifying binary output for log files; see Section 7.6.4, "Exporting Binary Log Files to Text Format", on page 45. Contact technical support before enabling extensive logging.
- If you install the Dialogic® system software on a FAT32 formatted hard drive, the RTF tool will generate an *Error after install attempting to launch* error message. The error message appears because FAT32 does not support file system access control. However, RTF will continue to function.
- When RTF logging is enabled, logs are stored in the *$(INTEL_DIALOGIC_DIR)\log* directory. You should take necessary precautions to ensure that the directory never fills.
- Occasionally, errors that appear in the RTF log can be ignored in certain circumstances. Check the Release Issues chapter in the *Release Update* for the Dialogic® software you are using. Such errors will be listed and explained in the Release Issues table.

7.6 rtftool Command

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

7.6.1 Pausing, Reloading, and Resuming RTF

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

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

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

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

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

7.6.2 Cleaning the RTF Log File

Use the `rtftool` command to clean or delete the RTF log file. To do so, use the following sequence of commands:

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

`rtftool clean [-f]`
Deletes the RTF log file. When the `-f` option is used, RTF will not ask the user for confirmation before deleting the log file.

`rtftool resume`
Resumes RTF tracing. Issue this command to resume tracing after RTF has deleted the log file.

7.6.3 Running RTF in Preservation Mode

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

```
rtftool preservation {on | off}
```

7.6.4 Exporting Binary Log Files to Text Format

The output of log files is set to text by default. For installations with high channel densities, or where all or most RTF trace levels are enabled, the volume of logging may result in an increased CPU utilization by the RtfServer executable as a result of the increased volume of log messages.

Test results have shown that generating RTF log files in binary format has less of an impact on CPU usage than text format. To change the format of the output log file, see the log_format attribute in Section 7.3.4, "Logfile Tag", on page 33.

To view binary log files, you must convert them into text format using the `rtftool export` command:

```
rtftool export [-d source_dir | -s source_file] [-f [dest_file] | -m
dest_dir]
```

By default, the name of the text format files generated by this command is *EXPORT-<RTF binary log file name>*. For example, if the binary format file is named *rtflog-LOCAL-20070306-15h09m26.506s.txt*, then the default name of the generated text format file is *EXPORT-rtflog-LOCAL-20070306-15h09m26.506s.txt*. You can override this behavior using the -f command line option.

The rtftool utility is a stand-alone program, and it is not necessary to have the Dialogic® software release installed on the system in order to convert RTF log files from binary to text format.

Note: When generating large binary files with RTF, do not split the single large binary file and then use the individual split files with the rtftool utility. Rtftool will not work with chopped binary files.

7.7 Example RTF Configuration Files

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

7.7.1 Example 1: Tracing disabled - RtfConfigWin.xml

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.

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

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

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

```
<Global>

</Global>

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

</RTFConfig >
```

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

The RTFConfig tag's trace attribute is set to 1 so tracing is enabled. The Global tag is empty, so there is no global configuration. For this example, tracing is only configured at the module level.

The path for the log file is *$(INTEL_DIALOGIC_DIR)/log*. Preservation mode is OFF, so the size and maxbackup attributes apply while the preserve_size and preserve_max_backups attributes are ignored. The maximum log file size is 1000 KB. The system maintains a maximum of 2 backup log files.

The only module configured for trace is *spwrvoice*. This means that all clients (devices) for this module are configured to trace ERR1 and WARN labels.

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

<RTFConfig trace="1" tracelocation=”TRACE_LOG” timestamp=”1”>

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

<Global>

</Global>

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

</RTFConfig >
```

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

The trace attribute of the RTFConfig tag is set to 1 so tracing is enabled.

The log file path is *$(INTEL_DIALOGIC_DIR)\log*. Preservation mode is ON, so the size and maxbackup attributes are ignored while the preserve_size and preserve_max_backups attributes apply. The maximum log file size is 300 KB. The system maintains a maximum of 10 backup log files (filenames appended with a *_p*).

The Global section is present in the configuration file, so all modules will have this global configuration. This configures “Entry” as a global label and “dxxxB1C2” is a global client for label “Exit”.

In the module section, there are two configurations for *spwrvoice* so the later configuration will take precedence. Since the state of *spwrvoice* is not specified, it takes default value “1”. Tracing is enabled for the module *spwrvoice* but only for the “WARN” label.

The client “dxxxB1C2” is also configured to trace in the *spwrvoice* module with the label “ERR1”. Therefore, the client “dxxxB1C2” in *spwrvoice* is traced for the label “ERR1”, even though it is disabled in the module section. All other clients of *spwrvoice* are configured to be traced for only label “WARN”, as “WARN” is the only label configured to be traced for the module.

The *faxntf* module is configured to trace “WARN” (from the module section) and “Entry” (from the global section).

The *dtintf* module is configured to trace “Entry” (from the global section). “dxxxB1C2” in *dtintf* is configured to trace “Entry” (from the global section) and “Exit” (from the global client). The other client “dxxxB2C1” is configured to trace “Entry” (from the global configuration) and “EXCE” (from the module client section). All other clients of this module are configured to trace “Entry” (from the global section) since these labels are configured for the module.

```
<?xml version="1.0" standalone="no"?>
<RTFConfig trace="1" tracelocation=”TRACE_LOG”>

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

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

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

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

   <MClient name="dxxxB1C2" >
      <MClientLabel name="ERR1"/>
   /MClient>
</Module>

<Module name="faxntf" state = "1">
   <MLabel  name="WARN" state = "1"/
</Module>

<Module name="dtintf">
   <MClient name="dxxxB2C1">
      <MClientLabel name="EXCE"/
   </MClient>

</Module>

</RTFConfig >
```

Telecom Subsystem Summary Tool (its_sysinfo) Reference 8

This chapter describes the Telecom Subsystem Summary Tool (its_sysinfo) and the information it collects.

8.1 Description

The Telecom Subsystem Summary Tool (its_sysinfo) provides a way to collect information about systems built using Dialogic 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, Dialogic® system software, and operational logs.

The its_sysinfo tool also enables you to collect baseline information about the system for quick review of configuration issues when determining system configuration consistency. This information is collected in a file, compressed, and archived as part of the complete system information collection in an archive file named *its_sysinfo.zip* (or a name you specify). If the installed system is configured in such a way that the baseline information is not available, the its_sysinfo tool will indicate "No Information Available."

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

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

- Section 2.2, "Collecting System Data to Diagnose an Application Failure or Crash", on page 14
- Section 2.3, "Creating a System Configuration Archive", on page 14

8.2 Command Line Interface

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

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

8.3 Graphical User Interface (Windows® only)

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

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

8.4 Information Collected by its_sysinfo

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

- **General System Information**
  - **Environment Variables** – information about the operating system environment variables
  - **System Event Logs** – See Table 3.
  - **Memory and Processor** – information about the platform's available and used memory and CPU type and number as of the time you executed the its_sysinfo tool
  - **Operating System** – information about the operating system's version, service pack, and language
  - **/proc/meminfo file** – Linux only
- **Information Specific to Dialogic® Products**
  - **Linux Package Info** – For Linux systems, its_sysinfo provides a Linux Package Info section that shows installed Dialogic RPM information and selections made from the Dialogic Install Menu.
  - **Windows® Package Info** – For Windows® systems, its_sysinfo provides a Windows Package Info section that shows the following information for the active system release and for the previously installed system release: release name and build number, build type, install location, install date, installed by, and installed features.
  - **Installed Devices** – its_sysinfo collects information about devices detected in Dialogic® system configurations and startup.
  - **Configuration Settings for Installed Devices** – its_sysinfo captures the stored values used by the configuration tool for Dialogic® system configurations and startup. For more information about the configuration tool, see the configuration guide(s) for the system release.
  - **Firmware Files and Versions** – its_sysinfo collects information about the files listing all firmware file names and version numbers.

- **FCD Files** – its_sysinfo collects information about the *.fcd* files used by the configuration tool for Dialogic® system configurations and startup. For more information about FCD files, see the configuration guide(s) for the appropriate software release.
- **PCD Files** – its_sysinfo collects information about the *.pcd* files used by the configuration tool for Dialogic® system configurations and startup. For more information about PCD files, see the configuration guide(s) for the appropriate software release.
- **CONFIG Files** – its_sysinfo collects information about the *.config* files used by the configuration tool for Dialogic® system configurations and startup. For more information about CONFIG files, see the configuration guide(s) for the appropriate 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, see the *Dialogic® Global Call Country Dependent Parameters (CDP) Configuration Guide*.
- **Build Information** – its_sysinfo collects information about the contents of the *buildinfo.ini* file used for the installed Dialogic® system. The *buildinfo.ini* file contains information about what is in the build of the software.
- **Board Memory Dumps** –
- **Event Viewer Data** – its_sysinfo collects information about the last events reported to the operating system and to the event logger from the Dialogic® system.
- **File Versions** –

The its_sysinfo tool also checks for the log files listed in Table 3 and adds them to the archive.

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

Table 3. Log Files Archived by its_sysinfo

Log Files	Windows®	Linux
OA&M files	$(SystemRoot)\System32\anm_debug.log $(SystemRoot)\System32\anm_trace.log %INTEL_DIALOGIC_DIR%\log\ClusterPkg.log %INTEL_DIALOGIC_DIR%\log\ClusterPkg.log.0 *or* $(SystemRoot)\System32\ClusterPkg.log* $(SystemRoot)\System32\confslot.log %INTEL_DIALOGIC_DIR%\log\ctbb*.log *or* $(SystemRoot)\System32\ctbb*.log dlgcInstall.log is in $(TEMP) %INTEL_DIALOGIC_DIR%\log\dlgcsyslogger.log %INTEL_DIALOGIC_DIR%\log\DM3AutoDump.log $(SystemRoot)\System32\dm3bsp.log $(SystemRoot)\System32\dm3fdspdll.log $(SystemRoot)\System32\frustatus.log %INTEL_DIALOGIC_DIR%\log\GenLoad.log %INTEL_DIALOGIC_DIR%\log\merc.log *or* $(SystemRoot)\System32\merc.log %INTEL_DIALOGIC_DIR%\log\ncm.ini %INTEL_DIALOGIC_DIR%\log\oam.log %INTEL_DIALOGIC_DIR%\log\Sctsassi.log	$(INTEL_DIALOGIC_DIR)/log/board*.log $(INTEL_DIALOGIC_DIR)/log/clusterpkg.log $(INTEL_DIALOGIC_DIR)/log/clusterpkg.log.* $(INTEL_DIALOGIC_DIR)/log/dlgsyslogger.log $(INTEL_DIALOGIC_DIR)/log/genload.log $(INTEL_DIALOGIC_DIR)/log/iptconf.log $(INTEL_DIALOGIC_DIR)/log/oam.log
Demos	%INTEL_DIALOGIC_DIR%\log\rgademo.log %INTEL_DIALOGIC_DIR%\log\Board[#].log	$(INTEL_DIALOGIC_DIR)/log/rgademo.log $(INTEL_DIALOGIC_DIR)/log/Board[#].log $(INTEL_DIALOGIC_DIR)/log/dlgrhdemo.log
RTF log files	<Logfile path>/rtflog*.txt <Logfile path> found in $(INTEL_DIALOGIC_DIR)\cfg\RtfConfigwin.xml or $(DLCFGPATH)\rtfconfig.xml	<Logfile path>/rtflog*.txt <Logfile path> specified in $(INTEL_DIALOGIC_DIR)/cfg/RtfConfigLinux.xml
	<Logfile path>/rtflog.txt <Logfile path> specified in $(DLCFGPATH)\rtfconfig.xml	<Logfile path>/rtflog*.txt <Logfile path> specified in $(DLGCFGPATH)/RtfConfigLinux.xml
CASTrace log files	$(INTEL_DIALOGIC_DIR)\log\CAStrace.log	$(INTEL_DIALOGIC_DIR)/log/CAStrace.log.*
Debug Angel files	$(INTEL_DIALOGIC_DIR)\bin\DebugAngel.log	$(INTEL_DIALOGIC_DIR)/log/debugangel.*
Pstndiag Trace log files	$(INTEL_DIALOGIC_DIR)\log\pstndiag.*	$(INTEL_DIALOGIC_DIR)/log/pstndiag.*

Table 3. Log Files Archived by its_sysinfo (Continued)

Log Files	Windows®	Linux
IP Protocol files	$(SystemRoot)\System32\gc_h3r.log *or* $(INTEL_DIALOGIC_DIR)\bin\gc_h3r.log $(SystemRoot)\System32\rtvsp1.log *or* $(INTEL_DIALOGIC_DIR)\bin\rtvsp1.log $(SystemRoot)\System32\sdplog.txt *or* $(INTEL_DIALOGIC_DIR)\bin\sdplog.txt $(SystemRoot)\System32\siplog.txt *or* $(INTEL_DIALOGIC_DIR)\bin\siplog.txt **Note:** This is obsolete. IP protocols support RTF.	$(HOME)/g*.log $(HOME)/rtvsp1.log **Note:** This is obsolete. IP protocols support RTF.
Dr. Watson dump and log files	$(USERPROFILE)\Local Settings\Application Data\Microsoft\Dr Watson\DrWtsn32.log	
Its_sysinfo logfile files	its_sysinfo.log	its_sysinfo.log

8.5 System Information Data Structuring

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

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