Intelligent Platform Management Interface (IPMI)

Conformance Test Suite (ICTS)

Condensed User's Guide

Order Number A00335-002

Revision History

Rev.	Date	Changes
0.1	08/99	Prototype Condensed ICTS User's Guide
0.2	12/99	Prototype 2 ICTS User's Guide
0.3	04/00	Incorporate ICMB and IPMB Tests and Hardware Installation
0.4	05/01	Add IPMB and ICMB transports and update batch mode setup
0.5	08/01	Added transports for IPMI 1.5, local Windows® 2000/Whistler for IA64, and SIOPROXY.EFI.
0.6	01/02	Updated text, figures and removed test descriptions.
0.7	04/03	SIOTRANS add command for KCS SMIC BT. Added TCL_EDITOR env variable definition.
0.8	11/04	Add for IPMI 2.0

PROHIBIT THE EXCLUSION OR LIMITATION OF LIABILITY FOR CONSEQUENTIAL OR INCIDENTAL DAMAGES, THE ABOVE LIMITATION MAY NOT APPLY TO YOU.

† Third-party brands and names are the property of their respective owners.

Contents

Figures

Tables

1 Overview

The Intelligent Platform Management Interface (IPMI) Conformance Test Suite (ICTS) User's Guide provides installation, configuration, project organization, and test suite usage information. It also describes the steps needed to prepare for testing, to run tests, and to manage tests in progress. Finally, the document lists the available tests and their individual usage characteristics.

1.1 Purpose of the IPMI Conformance

The IPMI 1.0/1.5/2.0 Conformance initiatives consists of a software framework within which you may execute a collection of predefined pass-fail tests for determining platform conformance with the IPMI 1.0/1.5/2.0 specification. The software framework includes a graphical user interface that provides configuration, loading, execution, and response access to the tests. The interface also allows recording of configuration file information for the host environment, the platform, the target, and user preferences. There are three different test methods consisting of the Conformance Test Suite (ICTS), Platform Management Bus (IPMB) and Chassis Management Bus (ICMB). Two additional testing methods Serial and LAN are available when IPMI1.0/1.5 Conformance Test suite is used with IPMI1.5 targets.

NOTE

The test suite does not imply or enforce compliance requirements.

1.2 Audience

This document supports firmware development engineers desiring IPMI 1.0/1.5/2.0 conformance. The document assumes familiarity with the IPMI 1.0/1.5/2.0 specification, engineering practices related to cross-platform development, and general testing theory. For additional information concerning the IPMI specification and cross-platform issues, refer to the Reference Documents section of this chapter.

1.3 Overview of Test Process

Use of the IPMI 1.0/1.5 Conformance Test Suite assumes a host environment and a system to be tested. Before use, you must install and configure the IPMI 1.0/1.5 Conformance Test. Configurations can be saved, so configuration may not be required to run tests. Once configuration files for a host and target are complete, testing involves invoking the testing framework, loading configuration files, loading tests, running tests, and analyzing the results.

1.3.1 ICTS Architecture Overview

The ICTS framework resides on a host system and provides a graphic interface for loading, running, and analyzing IPMI conformance tests. Once installed and configured, the system provides a set of automatically loaded tests, access to a test library, and access to library support modules for communications with transport modules.

The test framework provides an interface for loading and running tests. Tests indicate their results in two different ways, as text messages in the framework message window for human consumption, and through internal status values which are passed up from child to parent tests and eventually to the framework itself. Parent tests utilize these status values to generate summary reports of child test results in the message window.

The FTF also loads and allows tests access to message procedures from the message library. The message library facilitates sending of messages to the firmware on the target system by providing the FTF and tests with a procedure interface to transport modules to allow accurate communication access to the target.

1.3.2 ICTS Architecture Layers

The FTF is a multi-layered tool primarily implemented in the Tcl scripting language. *Transport Modules* may be either C or Tcl. The following list contains the names of the major components and descriptions of their function.

Base Framework – An application program. In combination with libraries and loaded *Test Modules* it is a complete test application. The base framework provides the user interface and the environment in which tests run.

Test Module – Conducts tests and reports a pass/fail result by conducting the test itself or by executing other test modules as sub-components. The *Base Framework* loads *Test Modules* in order to create a complete test application program.

Message Library – A set of transport-independent message passing routines using services provided by loaded *Transport Modules.* The Message Library selects the default transport in the absence of an explicit requested by the caller.

Transport Modules – A set of loadable modules, one module per transport, that provides primitives for target interface communication. The module contains additional administrative primitives not used directly by the *Test Modules*, but which are used by the *Base Framework* on behalf of the *Test Modules*.

In addition to these components, the framework includes several *Companion Libraries* that provide various utility services to the *Test Modules*.

Figure 1-1 shows the relationship between the various components of the ICTS firmware test framework.

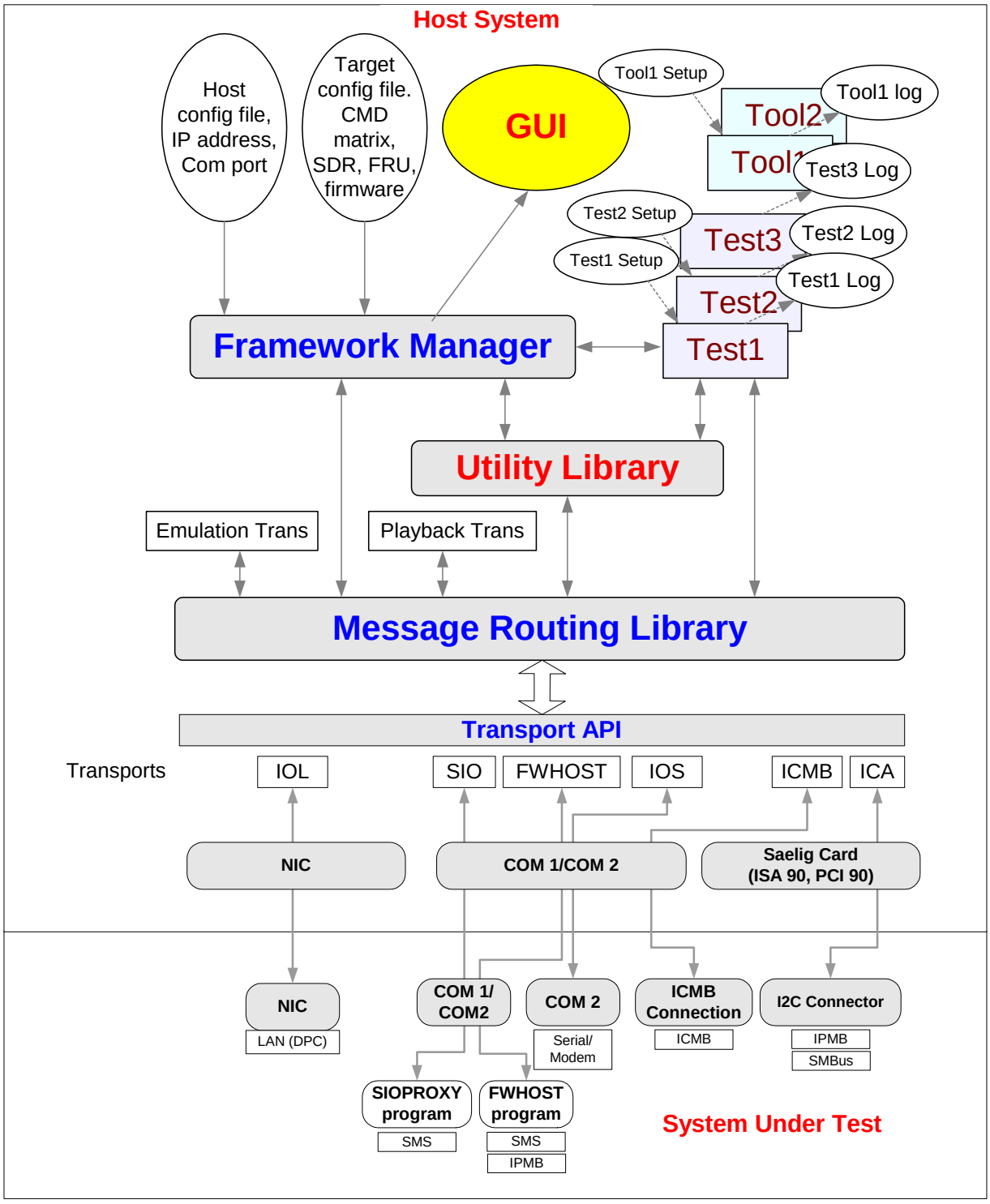


Figure 1-1. Firmware Test Framework Architecture

1.3.3 Installation and Configuration Overview

Before using ICTS, you must install and configure the system. For detailed instructions on installation and configuration, refer to the "Installation and Configuration" chapter of this manual. Generally, installation and configuration involve the following steps:

1. Developing a system to be tested.
2. Installing the Intelligent Platform Management Interface Test Suite software.
3. Checking the installed directory layout.
4. Configuring the host environment.
5. Customizing the Batch Files.
6. Configuring the platform.
7. Configuring the target.
8. Optionally creating user configuration files.

1.3.4 Test Session Process Overview

After installation and configuration, normal testing involves the following steps:

1. Starting the framework.
2. Loading configuration files.
3. Loading tests.
4. Running the tests.
5. Analyzing the results.

1.4 Conformance Scope

These tests provides pass-fail verification of IPMI 1.0/1.5 specifications for function module access commands and data storage requirements for SDR, SEL and FRU. The test suite does not test the internal workings of the target system.

1.4.1 ICTS Supports

ICTS supports the following IPMI conformance requirements:

- Management controllers, including BMC, should implement IPM functions.
- BMC should implement any mandatory functions such as IPM device, system interface, SDR repository Watchdog timer, Event Receiver, SEL Interface, and Internal Event Generator.
- Each mandatory function should be present and implemented as defined by the IPMI specification.
- Any conditionally mandatory IPMI 1.0/1.5 command should be present and implemented as defined by the IPMI specification if the specified condition from IPMI 1.0/1.5 is met or present.
- Any optional IPMI 1.0/1.5 commands should be implemented as defined by the IPMI specification if the command is present.
- Any data stored in SEL or SDR should follow the format specified in the IPMI 1.0/1.5 specification.

- Mandatory logical devices should be implemented, such as BMC, SEL and SDR.
- If the IPMB is present, the mandatory IPMI commands should be transferred via it (unless a command is specified as mandatory for only the system interface).
- If the IPMB is present, the additional mandatory IPMB messaging support commands in the BMC must be implemented as defined by the IMPI specification.
- If the IPMB is present, it is highly recommended that the Initialization Agent should be implemented. If it is implemented then it must be done as defined by the IPMI specification.
- If LAN or Serial Channels are present, the mandatory IPMI Commands should be transferred via the respective interface using one of these channels for each LAN and Serial Channel (unless a command is specified as mandatory for only the system interface).
- The platform must provide one of the system interfaces, either KCS, SMIC or BT.

1.4.2 ICTS Does Not Support

ICTS does not support the following test classes:

- Target platform-specific functions are neither tested nor verified.
- The following list of functions are example of functions that do not fall under the ICTS scope:
  - OEM-specific functions. Network function code 30h-3Fh
  - Actual data field in OEM SEL Record Type C0h-DFh, Type E0h-FFh
  - Actual data field in OEM SDR, SDR Type 0C0h
  - Firmware commands. Network Function Code 08, 09
- ICTS does not measure target platform quality.
- ICTS does not do stealth tests and exception tests, unless explicitly mentioned in the IPMI 1.0/1.5 specifications.
- ICTS does not detect management controllers in the target system. It only takes target system information from the SDR.

For additional information on the IPMI 1.0/1.5 specification, refer to the specification listed in the "Reference Documents" section of this chapter.

1.5 Reference Documents

In addition to the information in this manual, the following documents may provide information useful to testing for IPMI 1.0/1.5 conformances:

- *Intelligent Platform Management Interface Specification* v2.0 Revision 1.0, © 2004 Intel® Corporation, Hewlett-Packard Company, NEC Corporation, and Dell Computer Corporation.
- *Intelligent Platform Management Interface Specification* v1.5 Revision 1.0, © 2001 Intel® Corporation, Hewlett-Packard Company, NEC Corporation, and Dell Computer Corporation.
- *Intelligent Platform Management Interface Specification* v1.0 Revision 1.1, © 1999 Intel® Corporation, Hewlett-Packard Company, NEC Corporation, and Dell Computer Corporation.
- *Intelligent Platform Management Bus Communications Protocol Specification* v1.0, rev. 1.2 © 2000 Intel® Corporation.
- *Intelligent Chassis Management Bus Bridge Specification v1.0*, rev. 1.2, © 2000 Intel® Corporation.

- *System Management Bus (SMBus) Specification, Version 2.0,* ©2000, Duracell Inc., Fujitsu Personal Systems Inc., Intel® Corporation, Linear Technology Corporation, Maxim Integrated Products, Mitsubishi Electric Corporation, Moltech Power Systems, PowerSmart Inc., Toshiba Battery Co., Ltd., Unitrode Corporation, USAR Systems.

1.6 Intel® Intel®

Glossary

This section contains terms used throughout this document. For additional information regarding terms, refer to the documents listed in the "Reference Documents" section of this chapter.

API

Application Program Interface

EFI

Extensible Firmware Interface

FTF

Firmware Test Framework

FRU

Field Replaceable Unit. A module or component that will typically be replaced in its entirety as part of a field service or repair operation.

Host

The machine executing the test, which may or may not be the same as the target.

ICTS

Intelligent Platform Management Interface (IPMI) Conformance Test Suite.

Interface

The local communication path on the target machine (I2C, SMS, etc.)

IPMB

Intelligent Platform Management Bus. Name for the architecture, protocol, and implementation of a special bus that interconnects the baseboard and chassis electronics and provides a communications medium for system platform management information. The bus is built on I^2C and provides a communications path between management controllers such as the BMC, the ICMB bridge controller, and the chassis management controller.

IPMI

Intelligent Platform Management Interface

SDR

Sensor Data Record. A data record that provides platform management sensor type, locations, event generation, and access information.

Target

The machine under test, which may or may not be the same as the host.

Transport Layer

The data communication path between the host and target machines (Local, RS-232, TCP/IP LAN, etc.).

UI

User Interface

IPM

Intelligent Platform Management

BMC

Baseboard Management Controller

SEL

System Event Log

Saelig card

A card providing the standard I^2C interface to access the IPMB

2 User Interface

The Intelligent Platform Management (IPMI) Conformance Test Suite (ICTS) and Chassis Management Bus (ICMB) consists of a host-executable graphical user interface to allow configuration, running of tests, and viewing of test results. The interface consists of a message window, a status bar, and menus. This chapter provides an overview description of the interface window and menus. For information on launching the test framework and on using the dialog boxes for configuration, testing and analysis, refer to the "Installation and Configuration" chapter of this manual and to the "Tutorial" chapter of the *Intelligent Platform Management (IPMI) Conformance Test Suite (ICTS) Developer's Guide*.

2.1 Windows

Each window or menu has a specific purpose. A description of each appears in this chapter, and every section contains a description of the window or menu and provides a graphic of its appearance during typical use.

Table 2-1 contains a list of the windows and their purposes.

Table 2-1. ICTS Graphical User Interface Windows

Screen Object Name	Purpose
Main Message Window	Provides space for viewing messages generated by configuration, loading, testing, and analysis. The menu bar at the top provides access to the other menus.
Menu Bar	Provides access to drop-down menus.
Status Bar	Displays messages from test or from the framework.

2.1.1 Main Window

The main ICTS window provides access to menus, system information, and test feedback. Figure 2-1 depicts the main window immediately after launch. Diagnostic messages for configuration file-loaded tests have scrolled past, and the status bar at the bottom of the window shows the framework's initial state.

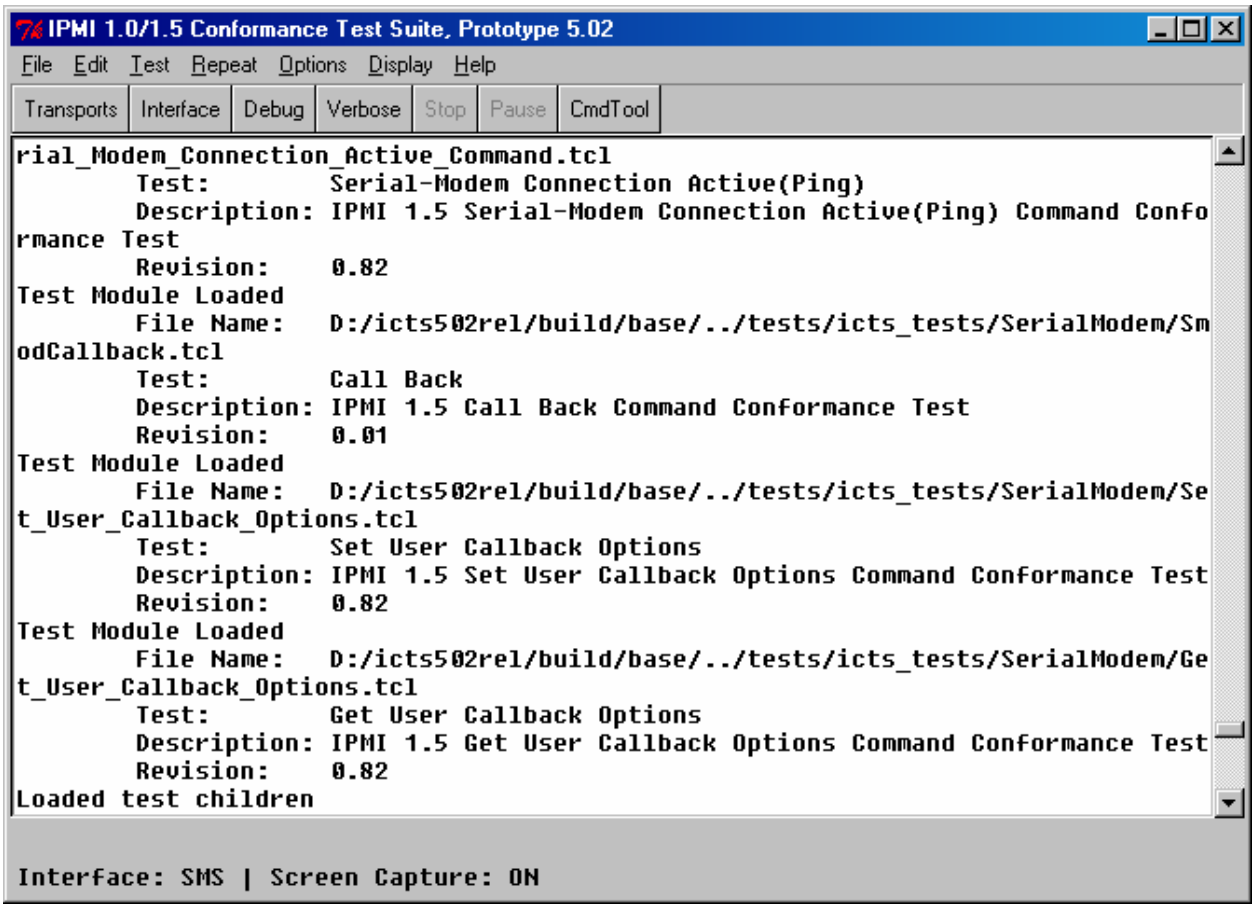


Figure 2-1. IPMI Conformance Test Suite Main Window

2.1.2 Status Bar

The Status Bar at the bottom of the main window shows general information about the user interface, including the target configuration, and the status of the screen capture utility. Additional information appears according to changes in the status of the framework.

2.2 Menus

Drop-down menus conform to Microsoft Windows NT† graphical user interface specifications. Each menu has a title in the menu bar at the top of the main window area. Menu commands are organized by type under menu titles. Click and hold on the menu title to access the menu items. Positioning the cursor over the menu command in the drop-down list and releasing the mouse button invokes the command item. Menu items followed by ellipses (. . .) invoke dialog boxes.

Every menu and menu command has a corresponding keystroke. Invoke drop-down menus by pressing the *ALT* key plus the underlined letter of the menu title. For example, to display the File drop-down menu, hold *ALT* and press *F*. Once a menu is invoked, pressing the underlined letter of any command listed in the drop-down executes that command. The *ESC* key cancels drop-down access.

This chapter contains a section describing each drop-down menu and its commands. Each section shows the screen during normal use. Table 2-2 shows the menu titles, their purposes, and the keystrokes for access.

Table 2-2. ICTS Graphical User Interface Menus

Menu Name	Keystroke	Purpose
File	ALT+F	Loading and saving configuration files.
Edit	ALT+E	Copying selected text.
Test	ALT+T	Loading, pausing, and stopping a test module.
Repeat	ALT+R	Repeating the last test run or the last tool used.
Options	ALT+O	Configuring verbose levels, message routing, interface type, data source, and micro controller type.
Display	ALT+D	Configuring window content and appearance.
Help	ALT+H or F1	Providing support information.

2.2.1 File Menu

The File drop-down menu gives access to dialog boxes that let you load configuration files for the host environment, the platform, the target, and the user. Additionally, the File menu offers access to a dialog box for modifying and saving user configuration files. Figure 2-2 shows the contents of the File drop-down menu:

Figure 2-2. File Menu

The following list contains the menu items offered in the File menu along with a function description of each:

- Save User Config—Saves the current user configuration.
- Exit—Exits from the program after presenting a confirmation dialog. Figure 2-3 shows the Exit dialog box:

Figure 2-3. Exit Framework Confirmation Box

2.2.2 Edit Menu

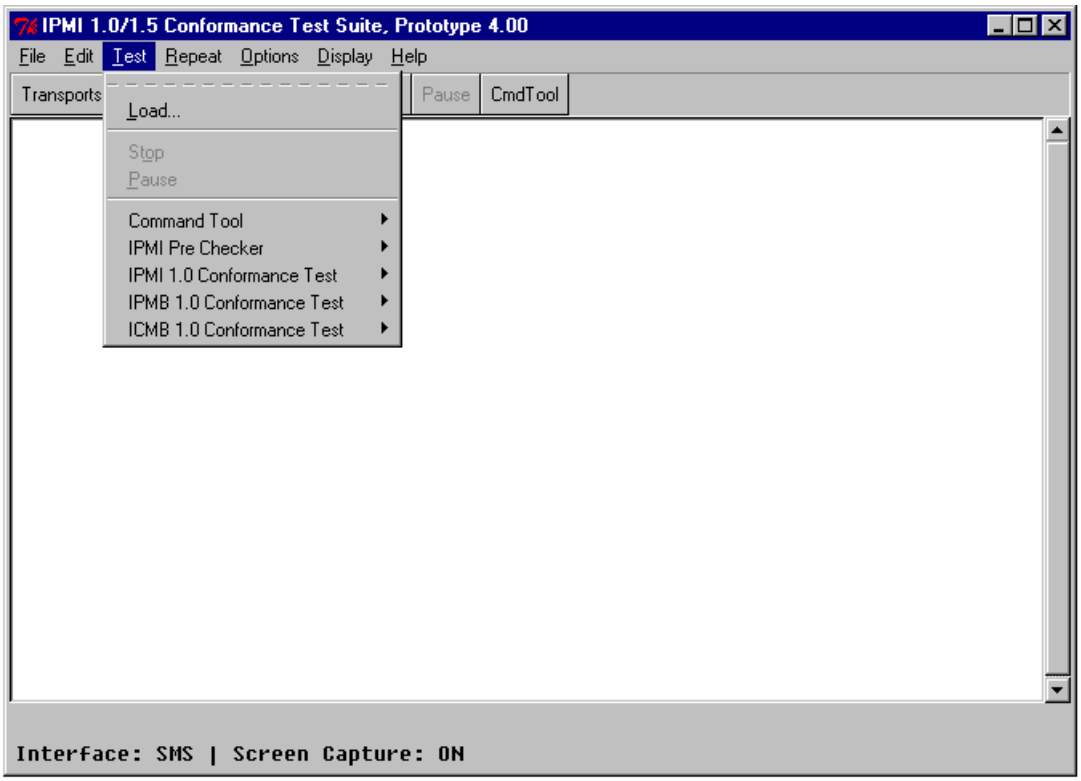
The Edit drop-down menu contains only the Copy command item. The Copy item remains unavailable, as shown in Figure 2-5, until a block of text is selected. Once text is selected, the Copy item appears as bold and enabled, as shown in Figure 2-4. Selecting the Copy item places the currently selected text in the system clipboard.

Figure 2-4. Edit Menu Copy-enabled

Figure 2-5. Edit Menu Copy-disabled

2.2.3 Test Menu

The Test drop-down menu has a unique characteristic. After a test is loaded, an additional menu item representing the newly loaded test appears in the IPMI 1.0/1.5 Conformance Test Suite drop-down menu. For additional information on using the Test drop-down menu to load, start, and stop tests, refer to the "Preparing to Run Tests" chapter of this manual.



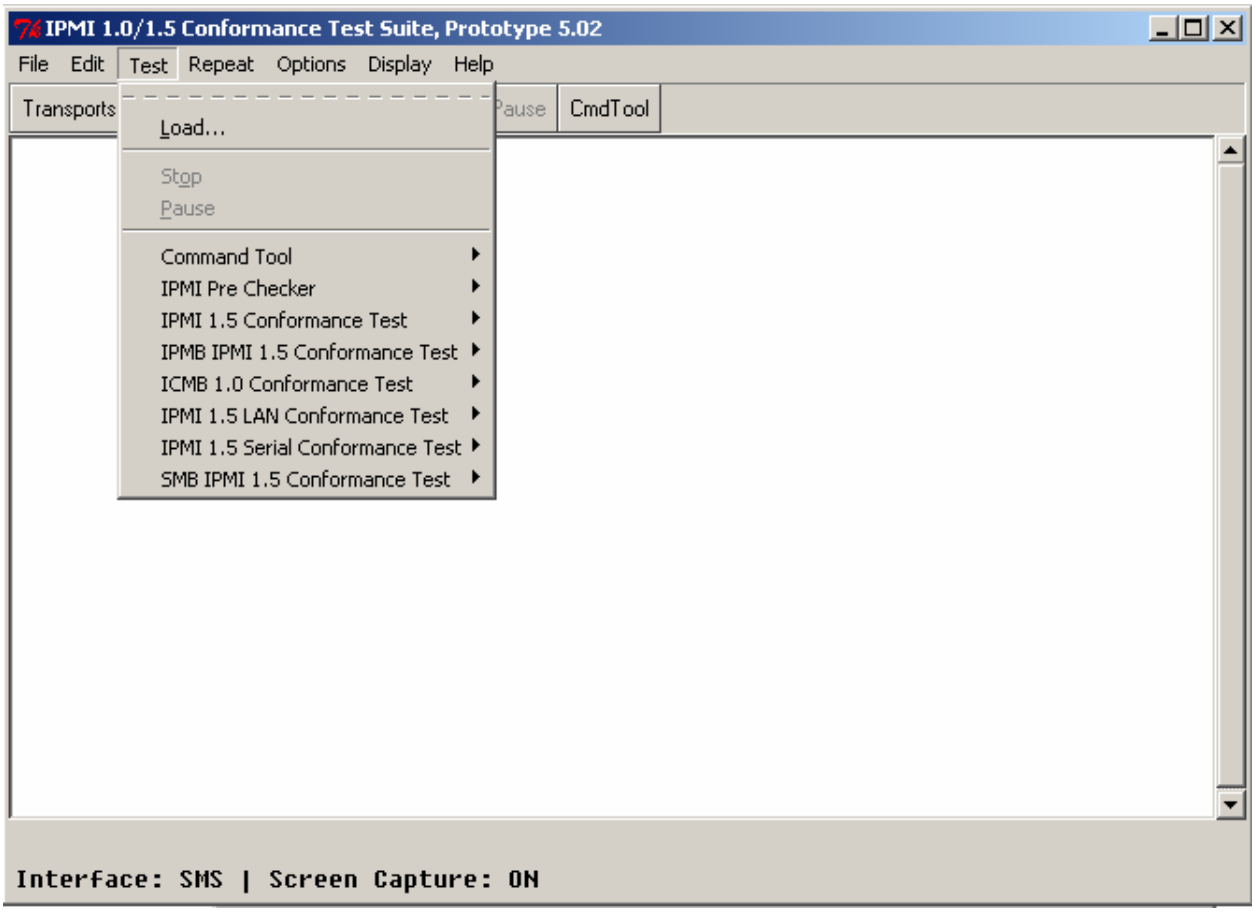


Figure 2-6. Test Menu

Initially, the Test drop-down menu contains these items:

- Stop—Stops the currently running test or tool, if possible. This menu item is unavailable unless a test or tool is currently running.
- Pause—Pauses the currently running test or tool, if possible. This menu item is dimmed unless a test or tool is currently running.
- IPMI 1.0/1.5 Conformance Test Suite—Provides access to additional drop-down menus for configuration of tests and access to test data. Figure 2-7 shows the contents of the IPMI 1.0/1.5 Conformance Test Suite dialog box. The dialog provides access to a series of drop-down menus through which you can access the automatically loaded test libraries provided by the framework. For additional information on the drop-down menus in this dialog, refer to the "IPMI Conformance Test Suite Sub-Menus" section of this chapter.

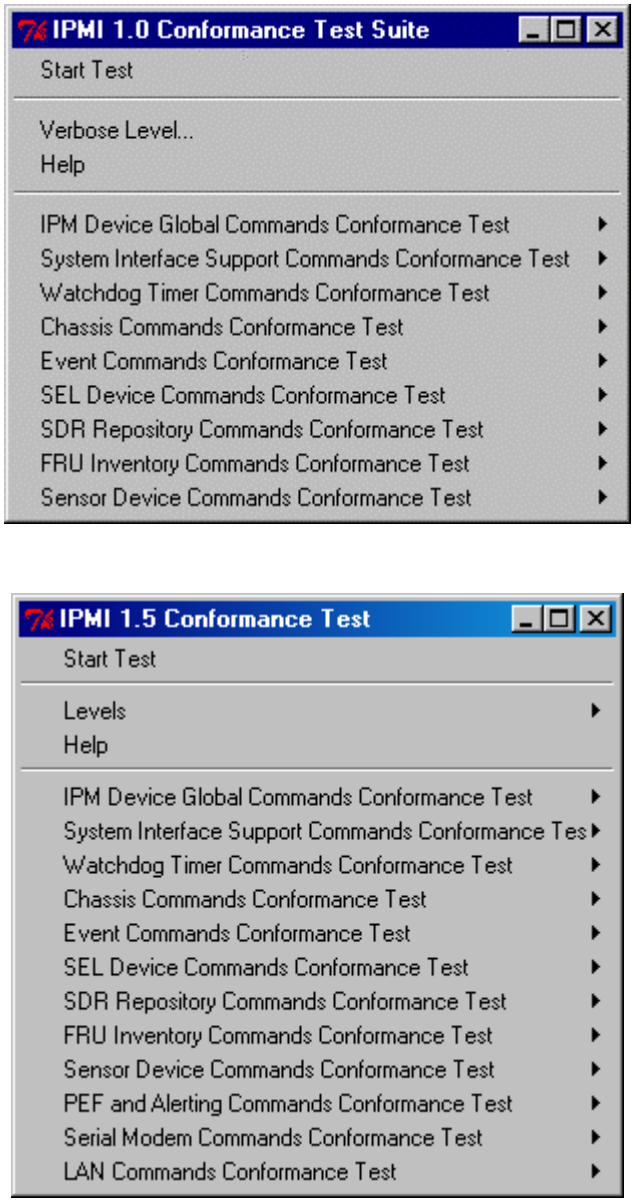


Figure 2-7. IPMI 1.0/1.5 Conformance Test Suite Drop-down Menus

2.2.3.1 IPMI Conformance Test Suite Sub-Menus

The IPMI 1.0/1.5 Conformance Test Suite submenus give access to the testing libraries provided by the test framework. Clicking Start Test on the IPMI 1.0/1.5 Conformance Test Suite menu runs the entire set of loaded tests. Selecting a child test from the test list sub-menu provides access to another test drop-down menu. The new menu appears identical to the one shown in Figure 2-7, except that the list of child tests changes. Each child test menu item provides access to automatically loaded tests. Selecting their respective Start Test item will run that test and its children.

Every test drop-down provides its own Start Test item, as well as Levels and Help items. Levels allows access to verbose level-setting. Help provides information available for the currently loaded tests.

Following the test hierarchy down to a test with no children in the sub-menu allows execution of a single test without initiation of child tests. Figure 2-8 shows the Get Device ID test menu. Get Device ID has no child tests, so there are no additional menu items.

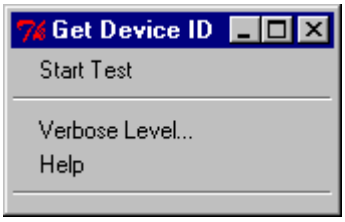


Figure 2-8. Standard Test Dialog

NOTE

The menus for a Custom Test are identical in form and content to the menus and actions for the standard tests described in this section.

2.2.4 Repeat Menu

The Repeat drop-down menu allows quick re-invocation of the last test run. Figure 2-9 shows the Repeat menu before a test has been run:

Figure 2-9. Repeat Menu

The Repeat drop-down menu contains these items:

- Last Test—Repeats the most recently completed test. This item remains dimmed until at least one test has successfully completed. During normal operation, the name of the last test run appears in this menu.

After a test has run, the Last Test item is enabled and shows the name of the last test run. Figure 2-10 shows the Repeat menu after the Get Device ID test has run:

Figure 2-10. Repeat Menu Showing Name of Last Test Run

2.2.5 Options Menu

The ICTS Options menu allows customization of the framework user interface behavior. For detailed usage information about the options presented by this menu, refer to the "Installation and Configuration" chapter in this manual.

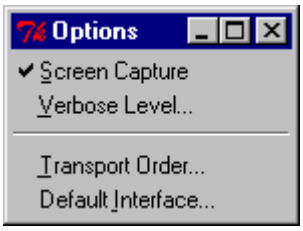


Figure 2-11. Options Menu

The Options drop-down menu contains the following items:

- Screen Capture—Toggles logging of screen messages to the host log file.
- Verbose Levels—Displays a cascading dialog for setting the global Verbose level. Figure 2-12 shows the dialog box:

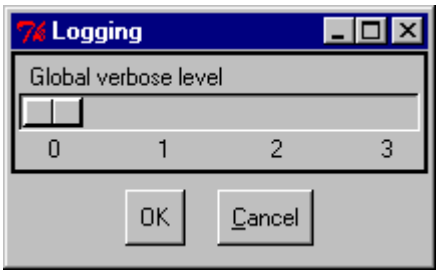


Figure 2-12. The Level Dialog

- Transport Order—Presents a dialog box for changing the priority order when multiple transport modules are available. Clicking on an item in the dialog promotes that item to the highest priority. Figure 2-13 shows the Transport Module Order dialog box containing two available transport layers:

Figure 2-13. Transport Module Order Dialog Box

- Default Interface—Presents a dialog box for selecting the default target interface. This dialog lists all open interfaces. Clicking on an item in the list selects that interface. Clicking OK makes the selected interface the default interface. The result appears in the status bar on the main window. Figure 2-14 shows the Default Interface dialog:

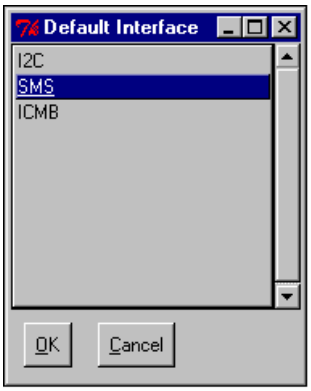


Figure 2-14. Default Interface Dialog Box

2.2.6 Display Menu

The Display menu items control the appearance of the ICTS interface window by allowing changes to the appearance of the graphical interface. Figure 2-15 shows the Display drop-down menu:

Figure 2-15. Display Menu

The Display drop-down menu contains these items:

- Clear—Clears the message window.
- Fonts—A cascading menu that provides access to font family selection and font size selection dialog boxes. Selecting the Default item returns the interface to its original state. Figure 2-16 shows the Fonts dialog box:

Figure 2-16. Fonts Dialog Box

Selecting the Family... item brings the Font Families dialog box up. Selecting a font family from the dialog and clicking the OK changes the font family for the user interface. Figure 2-17 shows the Font Families selection dialog box:

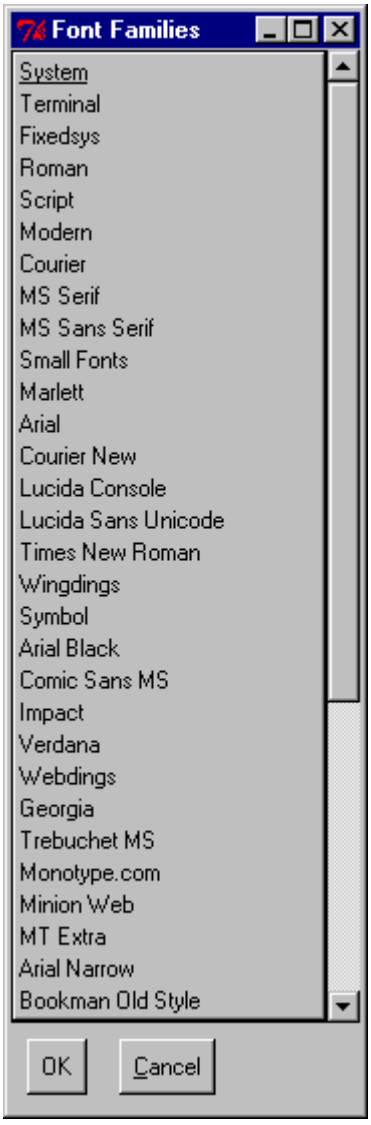


Figure 2-17. Font Families Selection Window

Selecting the Size... item brings the Font Sizes dialog box up. Selecting a size from the dialog and clicking the OK changes the font size for the user interface. Figure 2-18 shows the Font Sizes selection dialog box:

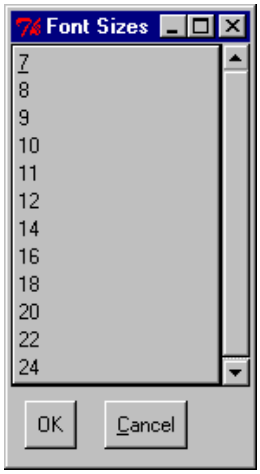


Figure 2-18. Font Sizes Selection Window

2.2.7 Help Menu

ICTS provides context-sensitive help through the F1 key and the Help drop-down menu. Shows the Help drop-down menu:

Figure 2-19. Help Menu

The Help menu contains these items:

- About—Presents a window displaying the program name, version number and copyright information.
- Introduction—Displays a brief how-to-get-started tutorial in the message window. The tutorial appears automatically if the framework starts without a host configuration file.
- Current Config—Displays the current value of all host, platform, target, and user configuration variables in the message window.
- Versions—Displays version number information for various components of the framework in the message window.
- Libraries—Displays a cascading list of loaded libraries. Selecting an item from the list displays help for that item. The list contents depend on the libraries currently loaded.
- Tests—A cascading menu containing an entry for each loaded test module. Selecting an entry displays information about the corresponding test in the message window.

3 Installation and Configuration

This chapter contains information on transport modules as well as hardware and software requirements for the IPMI Conformance Test Suite (ICTS). It also contains ICTS installation and configuration procedures along with the hardware and software requirements for ICTS.

3.1 ICTS Operating Requirements

Your Intelligent Platform Management Interface (IPMI) Conformance Test Suite (ICTS) requires hardware and software components. In addition, you will need to access the Intel® IPMI Web site and download an executable .zip file containing installation and configuration files. This section contains descriptions of the components you will need to install and configure the test suite.

3.1.1 Hardware Requirement

ICTS requires access to Tcl/Tk software, transport modules, configuration files, and hardware. Before attempting to install the Test Suite, check your hardware to be sure the minimum requirements listed below are met:

- IA microprocessor
- 24MB of available system memory (32MB is recommended)
- 20MB of disk space
- VGA adapter and monitor capable of providing 16 color, 640x480 graphics mode
- 101-key or compatible keyboard
- Mouse (optional)

3.1.2 Software Requirements

Before installing the ICTS software, be sure the following requirements are met:

- Installed Windows† 95, Windows 98, Windows NT 4.0, Windows® 2000 or Windows® 2000 Whistler for Itanium™ brand servers.
- Tcl/Tk 8.3.0 through 8.3.9. (The use of different versions will cause unpredictable behavior.)

3.2 Transport Modules

Transport modules are Tcl or C routines that allow messaging across a particular transport layer. Transport modules share an API. Some modules do not implement every standard routine, and some modules contain non-standard routines. One transport module must be present to run tests.

WARNING

Avoid direct calls to transport module routines. Use the Message Library or higher-level libraries described in the *Intelligent Platform Management Interface (IPMI) Conformance Test Suite (ICTS) Developer's Guide.*

3.2.1 General Requirements

Transport modules are passive Tcl or C routines that require no other procedure or data definitions before loading. They do not initiate interactions with their environment. Transport routines require a Tcl interpreter, but do not require the ICTS environment to load. Once loaded, the ICTS framework automatically determines the supported commands and interfaces of a loaded transport module.

3.2.2 Local Transport Module

The LOCTRANS implementation is a driver *imbdrv.sys* that can be installed under Windows NT 4.0 with service pack #4 or greater. The driver can be installed under Windows® 2000. The driver can be installed under Windows® 2000 Whistler for Itanium™ brand servers, though the driver handles the Windows® 2000 Whistler environment the ICTS application runs under the 32-bit application environment. If this driver is installed on a non IPMI platform, it may result in malfunctioning of the system. Eg. Sytem might not boot up after installation The following Table 3-1 lists the cross-layer communication and the LOCTRANS implementation for the Local Transport Module:

Table 3-1 LOCTRANS Implementation

Transport Characteristic	LOCTRANS Implementation
Software implementation	Four files: `loctrans.tcl`, `imb.dll`, `imbapi.dll`, and `imbdrv.sys`. The latter two files are part of the standard server-management software stack
Logical transports	None
IPMI interfaces	SMS
IPMI interface nodes	None
Raw interfaces	None
Non-IPMI interfaces	None
Target hardware requirements	KCS or SMIC interface
Target software requirements	The `imbdrv.sys` driver must be installed under Windows® NT 4.0 with service pack #4 or Windows® 2000
Host hardware requirements	Must be the same physical machine as the target
Host software requirements	None
Host operating system	See OS requirement under “Software Requirements”
Tcl/Tk version	Tested with 8.3.0 through 8.3.9
`topen` portid	None
`topen` options	None
`topen` limitations	Only one open port is allowed at any given time
`tsend/tget` limitations	Limited only by available memory
Incoming queue limitations	Limited only by available memory
Timeout implementation	The timeout value is the ISA timeout between the IMB driver and the firmware. The default is 300 ms and it may be changed with `ttimeout`. The timeout parameter to `tget` and `trawget` has no meaning.
Option procedures	`Terror, tdebug`
Non-standard procedures	None

3.2.3 The Serial I/O (SIO) Transport Module

The following Table 3-2 lists the cross-layer communication characteristics and SIOTRANS implementation for the SIO Serial Transport Module:

Table 3-2 SIOTRANS Implementation

Transport Characteristic	SIOTRANS Implementation
Software Implementation	Single file, siotrans.tcl
Logical Transports	None
Interface Types	IPMI, CFG
Interfaces (IPMI)	SMS
Interfaces (CFG)	SIO (see below)
Interface Nodes	None.
Raw Interfaces	None
Target Hardware Requirements	Null modem cable connected to COM port.
Target Software Requirements	Must be running SIOPROXY.EXE under DOS or SIOPROXY.EFI under the EFI Shell.
Host Hardware Requirements	Other end of null modem cable connected to a COM port.
Host Software Requirements	None.
Host Operating System	Win9x, WinNT, Win2000
Tcl/Tk Version	Tested with 8.3.0 through 8.3.9
topen portid	COM port name (COM1, COM2, etc.). No default.
topen options	None
topen limitations	None
tsend/tget limitations	None.
Incoming queue limitations	None.
Timeout Implementation	Default value is 1 second, but set in I_tranget.tcl to 6 seconds. May be changed with ConfigTool (see below).
Optional Procedures	tdebug, tversion
Non-standard Procedures	None.

SIO Interface

The “SIO” interface is a non-IPMI interface that can be used to send configuration commands to the SIOPROXY program using `tsend` and `tget`. A cooked SIO command for `tsend` has the form:

```
<configuration cmd> <data1> <data2> ... <dataN>
```

The commands are listed in the Table 3-3. The response format from `tget` for the SIO interface is:

<data1> <data2> ... <dataN>

Table 3-3. SIO Configuration Commands

Code	Command	Request Data	Response Data
00h	Hello	-	Byte 1: Completion Code Byte 2: Version Number (LSB) Byte 3: Version Number (MSB)
01h	Exit Program	-	Byte 1: Completion Code
02h	Verbose Level	Byte 1: New Verbose Level (optional)	Byte 1: Completion Code Byte 2: Previous Verbose Level
03h	Debug Level	Byte 1: New Debug Level (optional)	Byte 1: Completion Code Byte 2: Previous Debug Level
04h	Text Message	Bytes 1-N: A null terminated text message to be displayed on the screen of the target machine.	Byte 1: Completion Code
05h	KCS test routine	Byte 1: KCS test number 0 – IBF flag test 1 – Command and Data bit test 2 – OBF flag test 3 – KCS S1/S0 bits test 4 – KCS status abort test 5 – KCS attention flag test Byte 2: IPMI version	Byte 1: Completion Code
06h	SMIC test routine	Byte 1: SMIC test number Byte 2: IPMI version	Byte 1: Completion Code
07h	BT test routine	Byte 1: BT test number Byte 2: IPMI version	Byte 1: Completion Code

Debug Levels

The debug levels for the SIOTRANS implementation of `tdebug` are as follows:

Level 0: No messages.

Level 1: Misc. debug messages.

Level 2: Print all outgoing and incoming raw data plus the level 1 messages.

The Verbose and Debug Levels SIO configuration commands can be changed using the SIO Tool. The tool can be loaded into ICTS from the tests\cfgtools folder and is called Sio Tool.tcl. Figure 3-1 shows the SIO tool:

3.2.4 Figure 3-1. SIO Tool

The IPMB Transport Module

The following Table 3-4 lists the cross-layer communication characteristics and ICATRANS implementation for the IPMB Transport Module:

Table 3-4. IPMB Implementation

Transport Characteristic	ICA90 and PCI90 (Saelig) Implementation
Software Implementation	Four files: `icatrans.tcl`, `icadrv.dll`, `icaio.sys` / `pci_i2c.sys`.
Logical Transports	None.
Interface Types	IPMI, CFG
Interfaces (IPMI)	I2C
Interfaces (CFG)	(CFG)
Raw Interfaces	I2C-Fully supported. The interface uses polled mode.
Target Hardware Requirements	None.
Target Software Requirements	None.
Host Hardware Requirements	ICA90(Saelig) ISA or PCI90(Saelig) PCI card with IPMB cable connected to the target system.
Host Software Requirements	None.
Host Operating System	WinNT, Win2000
Tcl/Tk Version	Tested with 8.3.0 through 8.3.9
`topen` portid	ICA90 card base address (default address is 0x310) or 0-based PCI90 card number.
`topen` options	ICA90 or PCI90 card's I2C slave address. Default slave address is 0x54. May be changed with ConfigTool.
`topen` limitations	One open port at a time.
`tsend/tget` limitations	Message queue handles up to 64(0x3F) messages.
Incoming queue limitations	Message queue handles up to 64(0x3F) messages.
Timeout Implementation	Default value is 1 seconds, but set in I_tranget.tcl to 6 seconds. May be changed with ConfigTool (see below).
Optional Procedures	`terror, tdebug`
Non-standard Procedures	None.

Debug Levels

The debug levels for the ICATRANS implementation of `tdebug` are as follows:

Level 0: No messages.

Level 1: Print all incoming cooked or raw data based on the type of call.

Level 2: Print all outgoing cooked or raw data based on the type of call, plus level 1 messages.

Level 3: Print all outgoing and incoming, i.e., level 1 messages plus level 2 messages.

3.2.5 The ICMB Transport Module

The following Table 3-5 lists the cross-layer communication and the ICMBTRANS implementation for the ICMB Transport Module:

Table 3-5. ICMBTRANS Implementation

Transport Characteristic	ICMBTRANS Implementation
Software Implementation	Single file, icmbtrans.tcl
Logical Transports	None
Interface Types	IPMI, CFG
Interfaces (IPMI)	ICMBTRANS: ICMB
Interfaces (CFG)	CFG
Interface Nodes	Corresponds to target ICMB Address specified in hexadecimal (e.g. 0x8019)
Raw Interfaces (IPMI)	ICMBTRANS: None
Target Hardware Requirements	ICMBTRANS: ICMB connector
Target Software Requirements	ICMBTRANS: Target must be configured for the specified COM port (Refer to configuring to a specific COM port for Quatech card)
Host Hardware Requirements	Other end of the spliced cable is connected to D9 connector of the RS485 converter. Tx (A) and Rx (A) must be wired together, Tx(B), Rx(B) must also be wired together
Host Software Requirements	None.
Host Operating System	Win9x, WinNT, Win2000
Tcl/Tk Version	8.3.0 through 8.3.9
`Topen` portid	COM port name (COM1, COM2, etc.) No default.
`Topen` options	Node Address specified in hexadecimal (e.g. 0x1000). May be changed with ConfigTool.
`Topen` limitations	One open port at a time.
`Tsend` limitations	If iface parameter is to ICMB interface, it must include the target node address. For example, ICMB:0x8019
`tget` limitations	Does not allow receiving broadcast packets.
Incoming queue limitations	Only packets targeted to the host ICMB node will be moved into the queue
Timeout Implementation	Default value is 1 seconds, but set in I_tranget.tcl to 6 seconds. May be changed with ConfigTool (see below).
Optional Procedures	Tdebug
Non-standard procedures	None.

3.2.5.1 Specifications and Characteristics

The ICMB transport module allows three levels of debug through `tdebug`. Setting debug to zero disallows messages. Setting debug to 2 sends all outgoing and incoming raw data to the port.

Debug Levels

The debug levels for the ICMBTRANS implementation of `tdebug` are as follows:

Level 0: No messages.

Level 1: No messages. (Reserved for future used.)

Level 2: Print all outgoing and incoming raw data plus the level 1 messages.

ICMB Raw Messages

The trawsend/trawget procedures are not implemented by the ICMB transport.

Cooked Message Format

The "ICMB" is an IPMI interface and hence the cooked message format of ICMB is identical to the IPMI cooked message format with the following interpretation of the NetFn field to support bridging:

All requests to the ICMB interface requests are bridged with one exception. Any requests that are sent using the 'Bridge' Net Function are assumed to be directed at the bridge device it self and will be routed appropriately.

<rsSA> <NetFn> <rsLUN> <Cmd> <Data1> <Data2> … <DataN>

where:

rsSA Responder's I2C Slave Address

NetFn Network function, a six-bit value

LUN Logical Unit Number, a two-bit value

Cmd Command Number

Data1…. Command Data

Cooked IPMI responses have the following format:

<Ccode> <Data1> <Data2> … <DataN>

where

Ccode Command Completion code

Data1… Command Response data

3.2.5.2 Setting up the RS485 converter hardware

Any RS485 hardware, such as a RS485 port converter or an RS485 PCI card could be used for the ICMB tests provided they meet the following requirements:

- The RS485 hardware must to present a serial (COMx) interface to system software.
- RS232 serial interface must be able to handle transmission at 19,200bps
- RS485 interface must support 2-wire protocol.

In order to reduce possibility of collisions, ICMB devices arbitrate for the bus. We do not require the RS485 test hardware to implement the arbitration scheme since this may require intimate knowledge of the hardware as well as modifications to the hardware and the OS driver software. The ICMB transport module and tests assume that the ICMB segment consists of only two devices: the target ICMB device and the RS485 hardware.

The ICMB transport module is verified to work on the RS485 port converter, 485OI9TB from B&B electronics (http://www.bb-elec.com). The wiring for the RS485 port converter is as follows:

1. The echo pin should be in the OFF position
2. Tx(A), Rx(A) pins must be wired together and connected to Tx/Rx(-) wire of the ICMB cable
3. Tx(B), Rx(B) pins must be wired together and connected to Tx/Rx(+) wire of the ICMB cable
4. RS485 side requires external power source. Connect the +12V DC and GND wires to appropriately.

3.2.5.3 Test startup sequence

The target system must be discovered before it responds to ICMB requests. The ICMB tests must execute the following sequence of steps before starting the actual test.

1. Find the ICMB address of the target system. This could be accomplished by reading the target ICMB bridge via SMS or by issuing GetICMBAddress ICMB command to broadcast address.
2. Send a SetDiscovered command to the target ICMB bridge device.

3.2.6 The SMB Transport Module

The following Table 3-6 lists the cross-layer communication characteristics and SMBTRANS implementation for the SMB Transport Module:

Table 3-6. SMB Implementation

Transport Characteristic	SMBTRANS Implementation
Software Implementation	Three files: smbtrans.tcl, icadrv.dll and icaio.sys / pci_i2c.sys
Logical Transports	None.
Interface Types	IPMI, CFG
Interfaces (IPMI)	SMB
Interfaces (CFG)	CFG
Interface Nodes	None.
Raw Interfaces	I2C-Fully supported. The interface uses polled mode.
Target Hardware Requirements	None.
Target Software Requirements	None.
Host Hardware Requirements	ICA90 ISA card or PCI90 PCI card (Saelig card) with IPMB cable connected to the target system.
Host Software Requirements	None.
Host Operating System	WinNT, Win2000
Tcl/Tk Version	Tested with 8.3.0 through 8.3.9
`topen` portid	ICA90 card base address (default address is 0x310) or 0-based PCI90 card number.
`topen` options	ICA90 or PCI90 card's I2C slave address. Default slave address is 0x54. May be changed with ConfigTool.
`topen` limitations	One open port at a time.
`tsend/tget` limitations	Message queue handles up to 64(0x3F) messages.
Incoming queue limitations	Message queue handles up to 64(0x3F) messages.
Timeout Implementation	Default value is 1 seconds, but set in I_tranget.tcl to 6 seconds. May be changed with ConfigTool.
Optional Procedures	`terror, tdebug`
Non-standard Procedures	None.

Debug Levels

The debug levels for the SMBTRANS implementation of `tdebug` are as follows:

Level 0: No messages.

Level 1: Print all incoming cooked or raw data based on the type of call.

Level 2: Print all outgoing cooked or raw data based on the type of call, plus level 1 messages.

Level 3: Print all outgoing and incoming, i.e., level 1 messages plus level 2 messages.

3.2.7 CFG Interface

The "CFG" interface is a non-IPMI interface that can be used to send configuration commands to the SIOTRANS, IPMBTRANS, ICMBTRANS and SMBTRANS transports using tsend and tget. A cooked CFG command for tsend has the form:

<Configuration cmd> <data1> <data2> … <dataN>

The response format from tget for the CFG interface is:

<data1> <data2> … <dataN>

The configuration commands and their parameters are shown in Table 3-7.

Table 3-7. CFG Configuration Commands

Command Name	Cmd	Request Data	Response Data
GET_TIMEOUT	0	None	Completion Code, milliseconds
SET_TIMEOUT	1	In milliseconds (1000 is 1 second)	Completion Code
GET_RETRY	2	None	Completion Code, Integer (Default is 3)
SET_RETRY	3	Integer	Completion Code
SET_SLAVE_ADDRESS	4	For ICMB Node Address – LSB, MSB or Hex byte for Slave Address.	Completion Code
GET_SLAVE_ADDRESS	5	None	Completion Code, for ICMB Node Address – LSB, MSB or Hex byte for Slave Address
SET_TARGET_ADDRESS	6	Node Address – LSB, MSB	Completion Code
GET_TARGET_ADDRESS	7	None	Completion Code, Node Address – LSB, MSB

The CFG configuration commands can be done using the Configuration Tool. If the command is not allowed by the transport then a message is printed out about invalid configuration command. The tool can be loaded into ICTS from the tests\cfgtools folder and is called ConfigTool.tcl. The Configuration Tool with the different commands that then tool can perform is shown in Figure 3-2.

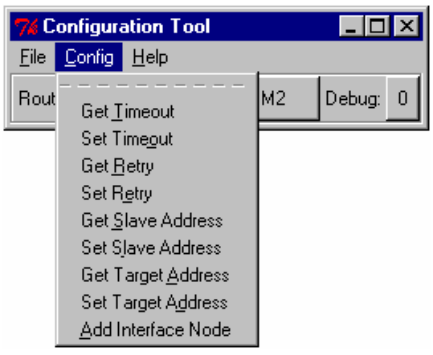


Figure 3-2. Configuration Tool with Configuration Commands

3.2.8 The Serial Transport Module

The following Table 3-8 lists the cross-layer communication and the IOSTRANS implementation for the IOS Transport Module:

Table 3-8. IOSTRANS Implementation

Transport Characteristic	IOSTRANS Implementation
Software Implementation	Single file, iostrans.tcl
Logical Transports	None
Interface Types	IPMI, CFG
Interfaces (IPMI)	IOSTRANS: IOS
Interfaces (CFG)	IOSCFG
Raw Interfaces (IPMI)	Yes, the transport fully supports a raw interface.
Target Hardware Requirements	Available serial connector on host computer. Null modem cable (for direct connect) or modem to modem hardware
Target Software Requirements	None
Host Hardware Requirements	Serial port supporting the basic mode serial interface with either direct connect and/or modem connect modes.
Host Software Requirements	None.
Host Operating System	Win9x, WinNT, Win2000
Tcl/Tk Version	8.3.0 through 8.3.9
`topen` portid	COM port name (COM1, COM2, etc.) No default.
`topen` options	See section 3.2.6.2
`topen` limitations	One open port at a time.
`tsend` limitations	None
`tget` limitations	None
Incoming queue limitations	None
Timeout Implementation	May be changed with the IOS configuration tool (iosTool.tcl see below).
Optional Procedures	tdebug
Non-standard procedures	None.

3.2.8.1 Specifications and Characteristics

The IOS transport module allows three levels of debug through `tdebug`. Setting debug to zero blocks displaying of any messages. Levels 1-3 show increasing levels of detail.

IOS Raw Messages

The trawsend/trawget procedures are implemented by the IOS transport. The format of messages is defined by the IPMI specification. The routine calling trawsend is responsible for all packet details including escaping and framing of the packet. Routines receiving data through Trawget must decipher the packet including removing framing and character escaping.

Cooked Message Format

The IOS transport is an IPMI interface and hence the cooked message format of IOS is identical to the IPMI cooked message format.

<dest> <netfn> <lun> <cmd> <data> <…>

where:

dest	Destination micro's I2C Slave Address
netfn	Network function, a six-bit value
lun	Logical Unit Number, a two-bit value
cmd	Command number
data	First command data byte
…	Additional data bytes

Cooked IPMI responses have the following format:

<Ccode> <Data1> <Data2> … <DataN>

where

Ccode	Command Completion code
Data1…	Command Response data

3.2.8.2 IOSTRANS Port Options

The configuration options available on opening the transport include:

- Baud
  This defines the baud rate setting of the testing unit's serial port. If connecting to a BMC via a direct connection, this value must match the BMC's baud rate setting. If connecting via a modem, the value must match the modem's baud rate setting.
- ModemInit
  This is a string used to initialize a host computer's modem if the transport is commanded to dial a connection to the BMC via a modem.
- ModemDial
  This is a string used to define a host computer's modem dial sequence if the transport is commanded to dial a connection to the BMC via a modem.
- PingWaitTime
  This defines how long the transport will wait for a ping before assuming that the BMC is not pinging.
- CommandTimeOut
  This defines the period of time in milliseconds that the transport will wait to receive a response from the BMC when the user requests to receive a response (via tget).
- PacketTimeOut
  This defines the period of time in milliseconds that the transport expects the BMC to finish an individual transmitting an individual packet. I.e. the transport will timeout if the time between a received message's frame start (0xa0) and the message's frame end (0xa5) exceed this value.
- ACKCheck
  This (value [0,1]) tells the transport whether to check for packet acknowledgments after sending a packet.
- WaitForPing
  This (value [0,1]) tells the transport whether to check for a ping message prior to sending a packet.
- RetryCount
  Specifies the number of times a packet is resent if the response is not received by the BMC. Note, this parameter's functionality is not implemented at this time.
- SequenceNumber
  This parameter allows one to configure the first sequence number to be used by the transport. Note, this number can range between 0 & 0x3f and is incremented and possibly wrapped to '0' prior to use.

These configuration options are sent to the transport upon opening the transport. They are defined in the Target_PortOptions(IOSTRANS) variable in the target configuration file. The parameters are defined in a string of the form "var1=val1 var2=val1…." The port options variable can be undefined or defined to an empty string if no options are desired, or can have as many pairs as the users wishes. The variable definition pairs are separated by spaces. There should not be any spaces

surrounding the equals sign. All of these parameters can be changed after the transport has been opened. See the next section for details about transport parameters and utility functions.

3.2.8.3 IOSTrans configuration options

The transport supports a rich list of configuration options available through tsend's IOSCFG interface. The options include:

isBMCPinging
This command will return data indicating whether a ping has been received and what type of ping (session active, session not active, mux switch set to system). Optionally, one may pass a string parameter with the value "noflush" to indicate that the routine should not flush the pings prior to checking for a ping.

isModemConnected
The command returns a true/false indication indicating whether the transport has an active connection through a modem.

dialModem
Executing this command causes the transport to dial a phone number using a modem. This command takes an optional string parameter defining the phone number to dial. If the optional parameter is not defined, the default string used is defined by the port options variable "ModemDial". This default can be manipulated with the setPhoneNum and getPhoneNum functions described below.

hangupModem
As one might guess, this causes the transport to hang up an existing modem connection.

muxBMC
This command causes the tranport to send two bytes (0x1B 0x28, "esc-(") out the serial port to cause the system to switch the mux to the BMC.

muxSystem
This command causes the tranport to send two bytes (0x1B 0x51, "esc-Q") out the serial port to cause the system to switch the mux to the system.

getModemState
This command returns an indication of the modem connection status.

closeAndOpenNewConnection
The command causes the tranport to close an re-open the serial port.

setRetryCount
This command is used to specify the number of retries the transport will attempt if a firmware command is sent, but a reply is not received. Use of the "retry count" variable is not implemented, so at this time, this command does not impact the transport's operation.

getRetryCount
This command is used to retrieve the number of retries the transport will attempt if a firmware command is sent, but a reply is not received. Use of the "retry count" variable is not implemented, so at this time, this command does not impact the transport's operation.

setBaudRate
This command takes a single parameter defining the new baud rate to use.

getBaudRate
The command returns the current baud rate.

setPhoneNum
This command defines the default phone number to call when making a connection through a modem.

getPhoneNum
The command causes the transport to return a sting indicating the current default phone number to use when connecting through a modem.

setAckChecking
This command take a true/false (1/0) parameter indicating whether the transport should check for BMC acknowledgements when sending messages to the BMC.

getAckChecking
This command returns the current state of BMC acknowledgment checking.

setPingChecking
This command takes a true/false (1/0) parameter indicating whether the transport should verify that the BMC is pinging before sending a message to the BMC.

getPingChecking
This command returns the current state of the transport's "ping checking".

getAckCount
Returns the number of BMC acknowledgements received since the transport was started.

getPortHandle
Return's the file handle to the serial port used by the transport.

setDebugLevel
This command accepts a single integer argument ranging from 0-3. The variable is used to increase or decrease the amount of debug information printed by the transport. The value '0' causes almost no information to be printed during the operation of the transport. The value '3' causes more information then most people can stand to be printed during the operation of the transport.

getDebugLevel
Returns the transport's current debug level setting.

setPingTimeout
The command expects a single integer parameter defining the maximum amount of time in milliseconds to wait for a ping message from the BMC. Note: if this value is shorter than the BMC's ping interval, then the transport will not be able to correctly determine if the BMC is pinging.

getPingTimeout
Returns the ping timeout.

setCommandTimeout
The command expects a single integer parameter defining the maximum amount of time in milliseconds to wait for a command response message from the BMC.

getCommandTimeout
Returns the command timeout.

setSequenceNumber
Defines the next sequence number for the transport to use. Note, because of the design of the transport the actual sequence number used will be 1 greater than the value specified by this command. Sequence numbers are incremented until the value 0x3f is reached, at which time the sequence number is reset to '0'. If a sequence number <1 or greater than 0x3f is sent, the next sequence number used will be '0'.

getSequenceNumber
Returns the current sequence number.

To send a configuration command to the transport, one should use the following examples:

```
puts [msendget IOSCFG [list setCommandTimeout 2000]]

puts [msendget IOSCFG [list getCommandTimeout]]
```

A graphical menu based tool has been written to expose all of these configuration commands. The tool is called iostool.tcl and is show in Figure 3-3.

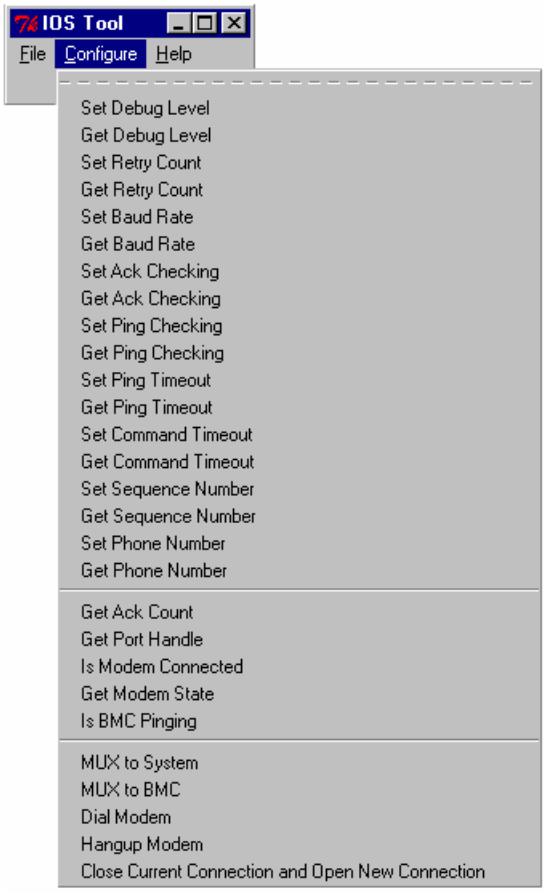


Figure 3-3. IOS Tool

3.2.9 The LAN Transport Module

The following Table 3-16 lists the cross-layer communication characteristics and IOLTRANS implementation for the IOL LAN Transport Module:

Table 3- 16. IOLTRANS Implementation

Transport Characteristic	IOLTRANS Implementation
Software Implementation	ioltrans.tcl, xtrans.dll, IOLTRANS.dll
Logical Transports	None
Interface Types	IPMI, IOLCTRL
Interfaces (IPMI)	IPMI Over LAN
Interface Nodes	None.
Raw Interfaces	None
Target Hardware Requirements	Firmware supporting IPMI 1.5 LAN channel.
Target Software Requirements	None
Host Hardware Requirements	Ethernet adaptor connected to a common network with the target.
Host Software Requirements	WinNT, Win2000
Host Operating System	Windows
Tcl/Tk Version	Tested with 8.3.0 through 8.3.9
Topen portid	An IP address in dot notation. For instance, 222.222.222.99
Topen options	A list of strings of the form, *-option <value>*. The option names can be entered as non-ambiguous partial strings (see table below).
Topen limitations	Currently limited to one session.
Tsend/tget limitations	Currently limited to one message transaction at a time.
Incoming queue limitations	Currently one.
Timeout Implementation	One second default. Timeout is specified as milliseconds. Timeout is on a per-message basis and defines how long the transport will wait for a response before retrying the same packet.
Optional Procedures	terror, tdebug
Non-standard Procedures	Transport provides an API for use in test scripts. The commands are defined in the following sections.

3.2.9.1 IOLCTRL Interface

The IOLCTRL interface supports non-numeric, string based, configuration parameters. Configuration commands are sent to the transport via this interface using tsend:

```
tsend <pd> IOLCTRL <-config_opt> <value> … <-config_opt> <value>
```

pd is the descriptor that was returned by topen and represents an instance of an IOLTRANS session object.

In a similar manner, configuration options can be passed to the transport when opening a new connection with the topen command:

```
topen <-config_opt> <value> … <-config_opt> <value>
```

`tsend and topen` are internal commands used by the framework. Test script developers will most likely not access these direction. Direct access to the configuration options below is provide through the API command, `iol_tconfigure`. This command is documented in the next section.

There are two components to IOLTRANS, one to handle IPMI 1.5 sessions and the other handles IPMI 2.0 (RMCP+) sessions. They both share many of the same configuration options; however, there are some differences as illustrated in the "Mode" column of the following table.

The Table 3-17 represents the configuration options and their required data.

Table 3-17. Configuration Options

Configuration Option	Request Data	Mode	Notes
-protoVer	[1.5|2.0]	1.5/2.0	This option is only writable via topen. This is indirectly set using the startup batch file and defining IOLOPT. Otherwise, it is read-only. When set to 1.5, the transport is operating per the IPMI 1.5 specification. When set to 2.0, it is communicating according to RMCP+ specifications.
-password	<password string>	1.5/2.0	The password the transport will use to calculate the authentication code in the session header. This option will zero extend or truncate the password to 16 bytes if needed.
-username	<user name string>	1.5/2.0	The username the transport will use to connect to the target.
-authtype	[none|straightpw|md2| md5]	1.5	The authentication type to use to establish the session. One of the strings listed to the left must be passed to this option.
-rakpAuth	[none|hmac_sha1| hmac_md5]	2.0	Sets the authentication algorithm used for RMCP+ session establishment.
-privlev	[callback|user|operator| admin]	1.5/2.0	The maximum Privilege Level the transport will request when connecting.
-managesession	[0|1]	1.5	Turns on/off the thread in the transport that occasionally pings the target and automatically reestablishes sessions if lost.
-timeout	<timeout value in ms>	1.5/2.0	Defines how long the transport will wait for a response for a given packet before resending it.

-retries	<number of retries>	1.5/2.0	The number of times to retry a packet after the initial packet doesn't receive a response.
-sessionSeqNum	<new sequence number>	1.5/2.0	This is the "inbound" sequence number from the BMC's perspective. Typically, this value should not be set unless testing this value is being done.
-sessionId	<new session ID to use>	1.5/2.0	This forces the transport to update the session ID in the session header without performing any additional steps. This value is also not typically changed, and the transport handles this value automatically.
-reconnect	[0|1]	1.5/2.0	Determines if the transport will assume a session is lost when a response is not received and attempt to connect a new session. Default is set to 0 (off).
-debug	<value 0+>	1.5	If set to >0, this causes the transport to send error, warning, and status messages to the FTF main window.
-payloadType	[ipmi_msg|sol|oem| open_sess_req| open_sess_rsp| rakp_1|rakp_2| rakp_3|rakp_4]	2.0	Sets the payload type field value in the RMCP+ session header.
-portNum	<integer>	1.5/2.0	Sets the management port value. Defaults to 623 in accordance with the IPMI specifications.
-sessActive	Read only [1|0]	1.5/2.0	Reports whether a session is currently active or not.
-sessMgrCount	Read only integer	1.5/2.0	Reports the current number of session manager objects instantiated in the transport.
-encryption	[none|aes_cbc_128]	2.0	Selects the RMCP+ encryption algorithm to use for the session.
-incrInbSeqNum	<integer>	1.5/2.0	Changes the sequence number for packets sent to the BMC. Used for testing only.
-integrity	[none|hmac_sha1| hmac_md5|md5]	2.0	Selects the RMCP+ integrity algorithm.
–lookupMode	[user_priv|name_only]	2.0	User_priv means both the request username and privilege will be used to lookup password key. This option value can be used combine with null username to enable "role only" login Nam_ only means user name alone is used to look up password key. Privilege level option is acts as a 'Maximum request privilege level' as in IPMI1.5

3.2.9.2 IOLTRANS Library Commands

In addition to supporting the IOLCTRL interface, the IOL transport implements several useful library commands that give the user a direct interface to these same features. These commands can be useful in scripting tests, tools, or in constructing a more versatile library. These commands are registered in the global namespace when the transport is loaded.

`iol_tconfigure <pd> <config_opt> <data> … <config_opt> <data>`

This command is constructed from the same code used to implement the IOLCTRL interface and accepts the same configuration options and values.

`iol_tconnect <pd>`

Causes the transport to connect the session object referred to by the descriptor to the target. In other words, the transport will attempt to establish a session using the specified session object.

`iol_tdisconnect <pd>`

When issued, the session object, pd, will send a Close Session request and terminate the session.

`iol_terror <ecode>`

Provides access to the ecode translator for the transport. Will return a short string describing the error value.

`iol_tgetDescriptorList`

Returns a Tcl list of descriptors. These descriptors can be used as parameters to the procedures described in this section.

In addition to supporting the standard test API for IPMI 1.5, IOLTRANS, when operating in IPMI 2.0 RMCP+ mode, provides the following additional commands for use in scripting. The following commands do not apply to IOLTRANS when operating in RMCP mode per the IPMI 1.5 specification.

`iol_tcreate`

This procedure creates a session manager, or connector object, inside the transport and returns a descriptor for the object. The new instance will need to be configured with an IP before rmcpp_tconnect is issued.

`iol_tdelete <pd>`

This command deletes the session manager object associated with the given descriptor.

`iol_warnAsFail <pd>`

This procedure configures the transport to send messages normally displayed as warnings to the ICTS window as fail messages.

3.2.9.3 Debug Levels

Currently, IOLTRANS supports only one tdebug level. When –debug is set to any value equal to or greater than 1, the transport will send messages to the main FTF window. These messages include error, warning, and status messages helpful in determining any problem the transport may encounter with a session.

3.3 Installation

The ICTS system includes components from several sources. This section contains instructions for acquiring the components necessary to install, configure, and run ICTS. Before attempting to install the system, refer to the "Hardware Requirement" and "Software Requirement" sections of this chapter to be sure your host system can support ICTS.

3.3.1 Downloading Tcl/Tk

Before installing ICTS, you must have Tcl/Tk version 8.3.0 through 8.3.9 on the host system. If you do not already have Tcl/Tk, complete the following procedure:

1. Log into the Web site: http://www.scriptics.com/software/8.3.html
2. Scroll down to the section titled “Downloading Binary Releases for Windows and Macintosh” and download the self-installing executable file for Windows 95, Windows 98, or Windows NT/2000.
3. Run the downloaded file on the host machine. Run any setup and configuration required in the readme file.

3.3.2 Downloading the Test Framework Suite

Once Tcl/Tk has been installed on the host system, you will need to download a copy of the test framework suite. To prepare for framework installation and to download the framework software, complete the following procedure:

1. Back up your existing framework installation.

⚠ CAUTION

Installing a new version of the framework over an old version will overwrite any files that you have customized. Creating a backup copy with WinZip† or another archiving utility will allow you to retrieve these customized files.

2. Log in to the Web site: http://developer.intel.com/design/servers/ipmi/
3. Download the self-extracting executable file containing the latest version of the framework.
4. Run the downloaded file in a temporary directory. Installation scripts, an archive file, and a text file named README.TXT appears in your temporary directory.
5. Read the README.TXT file.

3.4 Installing Firmware Test Framework and Configuration Files

Installation of the Firmware Test Framework requires the hardware and software described in the "ICTS Operating Requirements" section of this chapter. Before attempting to install the Firmware Test Framework, be sure the hardware and software meet the requirements.

Unless directed to do otherwise by the `README.TXT` file extracted from the downloaded archive file, install the firmware test framework on your PC using the following procedure:

1. Execute the extracted file `setup.tcl` using one of the following two methods:
   - *Click* (or double-click) on file `setup.tcl` from the Windows File Manager or Windows Explorer†.
   - Enter the following command at a DOS command prompt: `wish83 setup.tcl`.
2. Answer the questions asked by the setup program.
3. Shutdown and reboot the system.

If you later want to uninstall the software this may be done with the following DOS command:

```
wish83 setup.tcl uninstall
```

3.5 Configuration

Firmware Test Framework configuration is automatic. This section describes default directory locations and locations of default configuration files. To simplify the testing process the tree structure of the installed file system allows the same command names to be used regardless of the type of test used.

3.5.1 Directory Layout

The installation procedure creates a number of directories. Some must appear as the installation process created them.

The following table lists the fixed directories and a description of their contents:

Table 3-9. Default Directories

Name	Description
Base\	Tcl files for the base framework
Bin\	Batch files for starting the framework
Dll\	Dynamic load libraries
Docs\	Documentation
Hosts\	Default directory for host configuration files
Libs\	Tcl files for framework libraries
Local\	Local configuration files – Always starts out empty, immune to being overwritten by future upgrades
Packages\	Tcl files for framework packages
Templates\	Templates and examples for configuration files and test modules
Logs\	Default directory for log files
Platforms\	Default directory for platform configuration files and command matrix files
Setup\	Default directory for test setup files
Targets\	Default directory for target configuration files
Tests\	Default directory for test modules and tool modules
Tmp\	Default directory for scratch files
Transports\	Default directory for transport modules
Users\	Default directory for user configuration files

3.5.2 Configuration Parameters

After installing ICTS, ICTS needs to be configured in order use the different transports. The following Table 3-10 shows configuration parameters. The config.txt file in the Docs\ directory also describes the configuration parameters.

Table 3-10. Configuration Parameters

Variable	Options	Set Value
ICTSROOT		Your ICTS directory name. Please use an 8.3 filename format (xxxxxxxx.yyy) instead of a long file/directory name.
ICTSSHELL		Your Tcl/Tk interpret name such as wish83.exe.
TCL_EDITOR		Full path name of TCL script editor. Use "/" instead of "\" in path. Define this variable to integrate edit & reload script function in ICTS.
IPMI	1.00 or 1.50 or 2.00	IPMI version, the default version is 1.00.

ICTSMODE	NORMAL (default) DEVELOPER	The ICTS mode you want to use. If set to NORMAL, the ICTS runs in the NORMAL mode, and the user cannot develop a self-test case. If set to DEVELOPER, the ICTS runs in the DEVELOPER mode, and the user can run the ICTS test case and also develop a self-test case.
ICTSTRANS	SIOTRANS LOCTRANS (default) NONE	The transport you want to use for IPMI testing using the KCS/SMIC interface or SMS interface. If you don't specify an option, the default value is LOCTRANS. To disable ICTSTRANS, set it to NONE. LOCTRANS works only on Windows NT† 4.0.
ICTSCOM	COM1 COM2 COM3 COM4	The serial port used in the SIOTRANS mode. If you select LOCTRANS, this variable will be ignored.
IPMBTRANS	ICATRANS	The transport you want to use for IPMB testing using the I2C interface. To disable the IPMB interface, unset the IPMBTRANS environment variable by executing: set IPMBTRANS=NONE
IPMBPORT	0x310 0 1	ISA default port etc. Port First PCI card determined by the system. Port Second PCI card determined by the system.
ICMBTRANS	ICMBTRANS	The transport you want to use for ICMB testing. To disable the ICMB interface, unset the ICMBTRANS environment variable by executing: set ICMBTRANS=NONE
ICMBCOM	COM1 COM2 COM3 COM4	The serial port to be used in the ICMBTRANS mode.
SMBTRANS	SMBTRANS	The transport you want to use for SMBus testing using the I2C interface. To disable the SMBus interface, unset the SMBTRANS environment variable by executing: set SMBTRANS=NONE
SMBPORT	0x310 0 1	ISA default port etc. Port First PCI card determined by the system. Port Second PCI card determined by the system.
IOSTRANS	IOSTRANS	The transport you want to use for Serial/Modem testing. To disable the IOS interface, unset the IOSTRANS environment variable by executing: set IOSTRANS=NONE
IOSCOM	COM1 COM2 COM3 COM4	The serial port to be used in the IOSTRANS mode.
IOLTRANS	IOLTRANS	The transport you want to use for LAN testing. To

		disable the LAN interface, unset the IOLTRANS environment variable by executing: set IOLTRANS=NONE
IOLPORT	XXX.XXX.XXX.XXX	The IP address of the target.
IOLOPT	<-option> <value>...	The IOLTRANS options. These options can be catenated in one string to set the IOLOPT variable. The LAN transport implements partial word completion on the parameters, so a sub-string of the option name or value can be provided if it is non-ambiguous. For instance "-priv admin" would be translated as "-privlev administrator".
	-privlev [callback|user|operator|administrator]	This option sets the privilege level that the transport will attempt to establish a session at.
	-authtype [none|md2|md5|straightpw]	This option sets the authentication type that the transport will use.
	-timeout <value in milliseconds>	This option determines how long the transport will wait for a response before resending the packet.
	-retries <value>	This option sets the number of times to retry a packet if no response is received.
	-debug [0|1]	This option turns off/on transport messages that appear in the FTF main window. These messages include error, warning, and status messages. This is off by default.
	-manageSession[0|1]	This option determines whether the transport will periodically send a packet to the target to keep the session alive. This is on by default.
	-reconnect [0|1]	This option determines if the transport will attempt to establish a new session if a response is not received to a packet after all retries have been completed. The default is on.
	-password <string>	This option sets the password string that the transport will use to connect with. It is null by default.
	-username <string>	This option sets the user name that the transport will use to connect with. It is null by default.
	-protoVer [1.5|2.0]	This option sets the transport in RMCP+ (IPMI 2.0) or RMCP (IPMI 1.5) session protocol mode.

		An example string of options and values follows, the user may use their own discretion in these settings: IOLOPT=-priv admin -auth md2 -timeout 3000 -retries 2 -debug 1 -manage 1 -reconnect 0
ADDMICRO	micro-controller micro-controller	Used to setup additional micro-controllers other than BMC. set ADDMICRO= micro-controller micro-controller etc.
MFGTESTONPASS	micro-controller password micro-controller password	Used to setup manufacture password on micro-controllers set MFGTESTONPASS=micro-controller password micro-controller password etc.

Examples: set ICTSROOT=C:\ICTS

set ICTSSHELL=C:\TCL\bin\wish83.exe

set ICTSMODE=DEVELOPER

set TCL_EDITOR=D:/Program files/GNU/vim/vim61/gvim.exe

set ICTSTRANS=SIOTRANS

set ICTSCOM=COM1

set IPMBTRANS=ICATRANS

set ICMBTRANS=ICMBTRANS

set ICMBCOM=COM2

set ADDMICRO= micro-controller micro-controller etc.

set MFGTESTONPASS=micro-controller password micro-controller password etc.

3.5.3 The Batch File

In directory `C:\FTF\bin`, the files `ICTS.BAT` (for normal mode), `ICTSDEV.BAT` (for developer mode), or `ICTSDEV5.BAT` (for developer mode IPMI 1.5) are batch files that start the framework in an ICTS configuration. If `ICTSROOT` is undefined, it attempts to define the root for itself. It also makes a guess at the name of the Tcl/Tk interpreter and sets the `ICTSSHELL` environment variable accordingly. Make sure the path is set to C:\Program Files\tcl\bin default or wherever Tcl/Tk is installed. To stop it from guessing, hardcode the values for these variables at the top of the batch file or elsewhere before running the batch file. (If you used the `setup.tcl` script for performing the ICTS installation, the `ICTSROOT` and `ICTSSHELL` variables are defined automatically.)

The other configuration parameters can be put into the environmental variables, put in `ICTS.BAT, ICTSDEV.BAT or ICTSDEV5.BAT,` or put into a separate batch file that is run before staring ICTS. To start ICTS use `ICTS.BAT, ICTSDEV.BAT, or ICTSDEV5.BAT` by running it from the command prompt. `ICTS.BAT, ICTSDEV.BAT, or ICTSDEV5.BAT` starts the FTF with the ICTS default host and user configuration files described in the fallowing

sections. The user configuration file in turn causes the default platform and target configuration files to be loaded along with the ICTS test modules.

3.5.4 Host Configuration Files

The host computer is where ICTS runs and where you will work. The default host configuration file resides at `C:\FTF\hosts`. In this directory you can find the default host configuration file for ICTS, `i_host.tcl`.

3.5.4.1 Host Configuration Variables

The following table lists the configuration variables from the `i_host.tcl` file. Each variable is accompanied by a description:

Table 3-11. Hostcfg.tcl Variables

Name	Description
Host_Name	Identifies host computer
Host_Description	More verbose identification of the host computer
Host_Revision	Revision number for this configuration file
Host_Transports	List of transport modules to load, in priority order. Do not include logical transports
Host_Ports	Array of lists indicating valid port IDs for the various transport modules, including logical transports
Host_TransDirs	List of directories in which to find transport modules
Host_PlatDirs	List of directories in which to find platform configuration files
Host_TargDirs	List of directories in which to find target configuration files
Host_SetupDirs	List of directories in which to find test setup files
Host_TestDirs	List of directories in which to find test modules
Host_UserDirs	List of directories in which to find/save user configuration files
Host_Dirs	Default for the above directory lists
Host_LogDir	Directory for writing log files
Host_LogFile	Name for the log file
Host_TmpDi	Directory for writing temporary files
Host_MailHost	Host name (or IP address) of an SMTP server

3.5.5 Platform Configuration

Platform configuration files normally contain generic information about the system under test. The configuration file makes that information available to the framework. For example, a Model 1000 platform configuration file provides the framework with information applicable to all Model 1000 systems.

Included with ICTS is a default platform configuration file named `i_platfm.tcl`. This is suitable for most generic IPMI 1.0/1.5 platforms. The platform configuration files is in the directory `C:\FTF\platforms`, which is the path assigned to the `Host_PlatDirs` variable in your host configuration file.

3.5.5.1 Platform Configuration Variables

The following table lists the configuration variables from the `i_platfm.tcl` file. Each variable in the table is accompanied by a description of its use, an example, and whether it is required:

Table 3-12. Platform Configuration Variables

Variable Name	Purpose
Platform_Name	Identifies target platform (SC450NX, L440GX+, and so forth)
Platform_Description	More verbose identification of the platform
Platform_Revision	Revision number for this file
Platform_IPMI_Ver	IPMI version number
Platform_Possible_uC	List of possible microcontrollers that could be on this platform

3.5.5.2 Target Configuration Definitions

This section describes the platform configuration file variables that define target characteristics. When the framework attempts to resolve relative paths, it tries the path stored in the target configuration file `Host_PlatDirs` variable first. If `Host_PlatDirs` is undefined, then the framework looks for the platform configuration file variable `Host_TargDirs`.

The following table lists the variables that apply to the target:

Table 3-13. Platform Configuration Target Variables

Variable Name	Purpose
Platform_[other]	Default values for Target variables.

3.5.6 Target Configuration

Target configuration files give the framework-specific test system information not captured in the platform configuration file. Most target configuration file settings have default settings assigned in a platform configuration file. If all default values appear in the platform file and they are appropriate to the specific system under test, the target file is unnecessary.

Included with ICTS is a default target configuration file named `i_target.tcl` for generic IPMI 1.0/1.5 systems. This will be suitable for ordinary use of ICTS.

3.5.6.1 Target Configuration Variables

Table 3-14 lists the configuration variables from the `targcfg.tcl` file. Each variable is accompanied by a description of its use:

Table 3-14. Target Configuration Variables

Variable Name	Purpose
Target_Name	Identifies target box
Target_Description	More verbose identification of the target
Target_Revision	Revision number for this file
Target_FW_Ver	Firmware version
Target_LogFile	Log file name
Target_Interfaces	List of interfaces available on the target
Target_SDR_Source	Determines source of SDR information
Target_Ports	Array of ports you want to open for each transport
Target_PortOptions	Array of port open options
Target_PortTimeouts	Array of port timeout settings in milliseconds
Target_BadRoutes	List of routes that are unavailable
Target_IPAddr Target_BroadcastIPAddr Target_IEEEAddr	Networking variables that may be defined as IP addresses or resolvable host names
Target_SDR_uC and Target_SDR_uC_Info	Provides SDR-like micro controller information if Target_SDR_Source is set to “Target_Config”
Target_BMC_SA	Specify the Target BMC slave address, default value is 0x20

3.5.7 User Configuration

The optional user configuration file provides user preference information to the framework. Included with ICTS is a default user configuration file name `i_user.tcl` suitable for the generic IPMI 1.0/1.5 system. The following section describes a method of capturing current user preferences in a new user configuration file. For additional information on changing user preferences for the framework, refer to the "Customizing User Settings" section of this manual.

3.5.7.1 Creating a User Configuration File

Once created, a user configuration file allows you to load the desired interface and test characteristics you have stored. A saved user file automatically loads the platform file, target file, and your suite of tests.

To create a user configuration file, complete the following procedure:

1. Start up the framework. For information on starting the framework, refer to the "User Interface" chapter of this manual or the "Tutorial" chapter of the *IPMI ICTS Developer’s Guide*.
2. Load the desired host, platform, and target configuration files. For information on loading configuration files, refer to the "Preparing to Run Tests" chapter of this manual.
3. Load the desired set of test modules. For information on loading test modules, refer to the "Running Conformance Test" chapter of this manual.

4. Using the Options and Display menus, set preferences for screen font, verbose level, and so forth.
5. Select the Save User Config... option from the File drop-down menu.
6. Place your user configuration files in directory `C:\FTF\users` or any other directory listed by the `Host_UserDirs` variable in your host configuration file.

4 Preparing to Run Tests

This chapter contains information on starting the test framework, exiting the framework, and customizing the framework. Before attempting to run the framework software for the first time, you should be familiar with the information provided in the "Installation and Configuration", and "User Interface" chapters of this manual.

4.1 Starting the Framework

Before starting the test framework, complete the installation and configuration instructions in the "Installation and Configuration" chapter of this manual. In normal use, the framework can be launched from a batch file, by command line.

4.1.1 Command Syntax

The invocation command line can be placed in a batch file, typed to the Run dialog box or typed to a prompt command line. The `ICTS.BAT` file is included with the framework and, it should be suitable for most cases. If you want to create custom batch files the information in this section may be useful.

The generic name for the Tcl/Tk interpreter command is `wish`. In the case of Tcl 8.3.0 through 8.3.9 (recommended for use with ICTS) the interpreter name is `wish83`. The interpreter allows any `.tcl` file as an argument. However, only the `ftf.tcl` file launches the framework.

Variable setting parameters to `ftf.tcl` override configuration file settings for those variables, and the order of the parameters does not affect precedence.

The syntax described below will bring the framework up on your workstation:

Wish83 `C:\FTF\base\ftf.tcl [`*varname*`[=[`*varvalue*`]]] ...`

Where:

Wish83	The Tcl/Tk interpreter version 8.3.0 through 8.3.9.
varname	A Tcl global variable.
varvalue	A Tcl global variable.

Each *varname/varvalue* combination may take one of three forms:

varname	Sets the named global variable to a value of one.
varname=	Deletes the named global variable.
varname=varvalue	Sets the named global variable to the specified value.

The equal sign syntax causes difficulty in some circumstances, namely in windows shortcuts and in parameters to DOS batch files. To avoid this problem, substitute a double-colon (“::”) for the equal sign.

4.1.2 Command Parameters

The Tcl global variables described in this section have special meaning only when defined as parameters to `ftf.tcl`. The variables described here may appear only once on the command line. The following table lists the special Tcl global variables by name, type, and purpose:

Table 4-1. Command Parameters

Name	Type	Description
Host	File Name	A host configuration file to load after starting the framework. Relative paths are resolved using %ICTSROOOT%\hosts. The default file name extension is .tcl.
Platform	File Name	A platform configuration file to load after starting the framework. Requires host. Relative paths are resolved using the Host_PlatDirs variable from the host configuration file. The default file name extension is .tcl.
Target	File Name	A target configuration file to load after starting the framework. Requires platform. Relative paths are resolved using the Host_TargDirs variable from the host configuration file. The default file name extension is .tcl.
Test	File Name	A test module to load after starting the framework. Requires platform and perhaps target. Relative paths are resolved using the Host_TestDirs variable from the host configuration file. The default file name extension is .tcl.
Setup	File Name	A setup file to assign to the test module specified with the test variable. Requires test. Relative paths are resolved using the Host_SetupDirs variable from the host configuration file. The default file name extension is .tcl.
User	File Name	A user configuration file to load after starting the framework. Requires host. Relative paths are resolved using the Host_UserDirs variable from the host configuration file. The default file name extension is .tcl.
Resource	File Name	A UNIX/X-Window style resource data base file.
Console	Integer	If non-zero, causes the wish console window to be opened.
Debug	Integer (0-3)	The global debug level.
Verbose	Integer (0-3)	The global verbose level.
Batch	Integer	If non-zero, enables batch mode.
Auto	Integer	If non-zero, auto-starts the test specified by the test variable. Requires test.
Help	Integer	If not-zero, causes help to be displayed containing an abbreviated form of the information in this table. (Also available as a Help menu option.)

4.1.3 Example Framework Invocation

Figure 4-1 shows the test framework user interface immediately after invocation. In the example shown, load information for the automatically loaded test libraries appears in the message window.

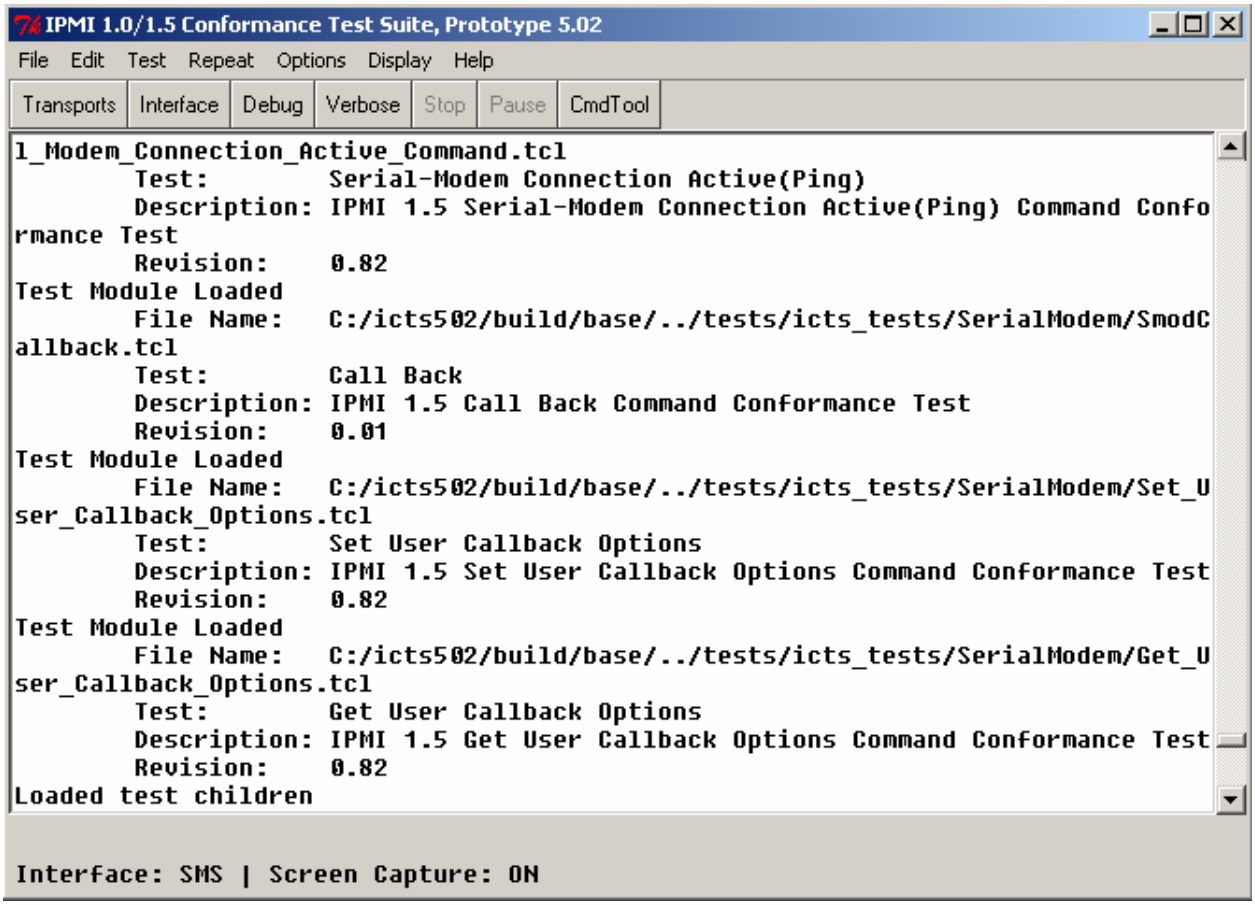


Figure 4-1. IPMI 1.0/1.5 Conformance Test Suite Main Window

4.2 Exiting the Framework

Once the testing session is complete, you can exit the framework by one of several methods. To exit the framework using the File drop-down menu, perform the following procedure:

1. *Click* the File menu. A drop-down menu appears. Figure 4-2 shows the File menu and the Exit selection.

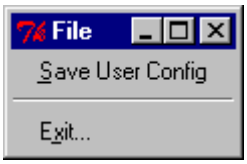


Figure 4-2. File Menu with Exit Selection

2. *Click* Exit... A confirmation dialog box appears. *Click* Yes to confirm or No to continue running the framework. Figure 4-3 shows the Exit Confirmation dialog box.

Figure 4-3. Exit Confirmation Dialog Box

In addition to closing the framework by using the Exit command, you can also exit by any of the following methods:

- Double-click the Framework icon in the upper-left corner of the framework window.
- *Click* the X button in the upper-right corner of the window.
- *Click* the Close option from the drop-down menu that appears when you click and hold on the icon in the upper-right corner.

In all cases, the confirmation dialog box shown in Figure 4-3 appears.

4.3 Customizing User Settings

After invoking the user interface, you may want to customize the framework for personal preferences in appearance or for repeated use for testing. The interface allows you to change the appearance of the user interface by sizing the window in the normal click-and-drag Windows NT manner. Additionally, you can change the message text appearance through the Display menu located in the menu bar at the top of the framework window.

The Display menu allows you to change the characteristics of your interface. You can change the font choice and size, and you can clear the settings to return to the default. Figure 4-4 shows the Display drop-down menu:

Figure 4-4. Display Drop-down Menu

In addition to changing the appearance of the interface, you can use the Options menu to toggle the screen capture feature on and off, set the global default level of verbose output, change the transport order, and select the default target interface. Figure 4-5 shows the Options drop-down menu:

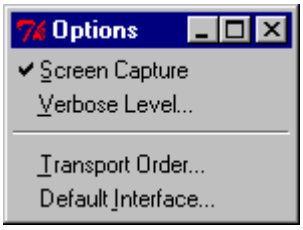


Figure 4-5. Options Drop-down Menu

4.3.1 Changing the Message Window Font

To select a new font, complete the following procedure:

1. *Click* Display in the menu bar.
2. *Click* Fonts... The Fonts dialog box appears. Figure 4-6 shows the Fonts dialog box:

Figure 4-6. Fonts Characteristics Dialog Box

3. *Click* Family... The Font Families selection dialog box appears. Figure 4-7 shows the Font Families selection dialog box.

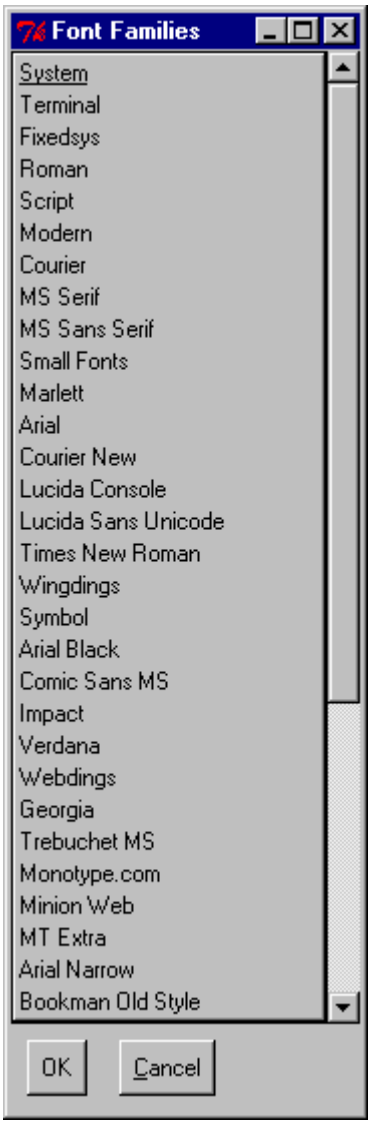


Figure 4-7. Font Family Selection Dialog Box

4. *Click* the font family you desire.
5. *Click* OK... The message window typeface changes to the newly selected font.

4.3.2 Changing the Font Size

To change the size of the font in the message window, complete the following procedure:

1. *Click* Display in the menu bar.
2. *Click* Fonts... The Fonts characteristics dialog box appears as shown in Figure 4-6.
3. *Click* Size... The Font Sizes selection dialog box appears. Figure 4-8 shows the Font Sizes selection dialog box.

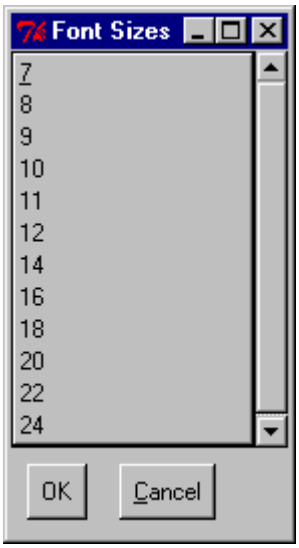


Figure 4-8. Font Sizes Selection Dialog Box

4. *Click* the font size you desire.
5. *Click* OK. The message window text changes to the newly selected size. You may not notice a change if the currently selected font family does not support the selected size. If nothing happens try a different size or a different font family.

4.3.3 Changing Verbose Output Level

To change the global or local verbose level, you essentially the same procedure. You can change the global output level by beginning the procedure from the Verbose Level item on the Options menu. To change the output level for a particular test and all of its children, begin the procedure from the Verbose Level item on the menu for a selected test.

Figure 4-9 shows the menu for the test, “Get Device ID.” Figure 4-10 shows the Options menu:

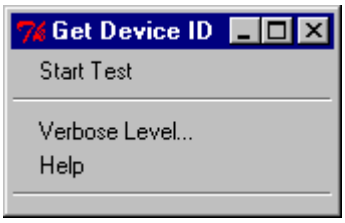


Figure 4-9. Test Menu Showing Levels Item

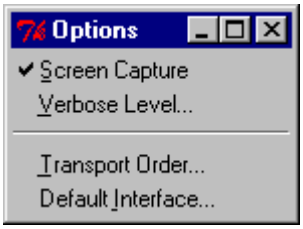


Figure 4-10. Options Menu Showing Levels Item

To change the output level, complete the following procedure:

1. *Click* on the Verbose Levels option from the appropriate menu. The Verbose Setting dialog box appears. Figure 4-11 shows the dialog box used for setting the verbose levels:

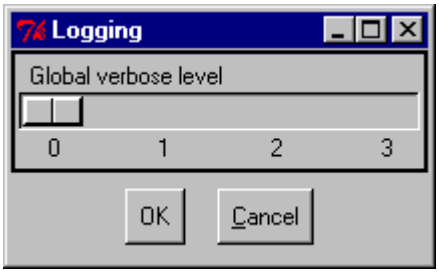


Figure 4-11. Global Verbose Dialog Box

2. Select the level of output you desire.
3. *Click* OK to enable the global verbose setting you have selected. *Click* on Cancel to return to the interface without changing the level.

4.3.4 Changing the Order of Transport Modules

If more than one transport module supports a given interface, such as SMS, the priority order feature of the message library for the transport modules determines which module gets responsibility for SMS messages. To change the priority order of the transport modules, complete the following procedure:

1. *Click* Options in the menu bar. The Options drop-down menu shown in Figure 4-5 appears.
2. *Click* Transport Order... The Transport Module Order dialog box appears. Figure 4-12 shows the Transport Module Order dialog box with two transport modules listed:

Figure 4-12. Transport Module Order Dialog Box

3. The next time the tests are run, the new high priority module handles messaging for the transport layer.

4.3.5 Changing the Default Target Interface

To change the default target interface, complete the following procedure:

1. *Click* Options in the menu bar. The Options drop-down menu shown in Figure 4-5 appears.
2. *Click* Default Interface.... The Default Interface dialog box appears. Figure 4-13 shows the Default Interface dialog box with one possible interface listed:

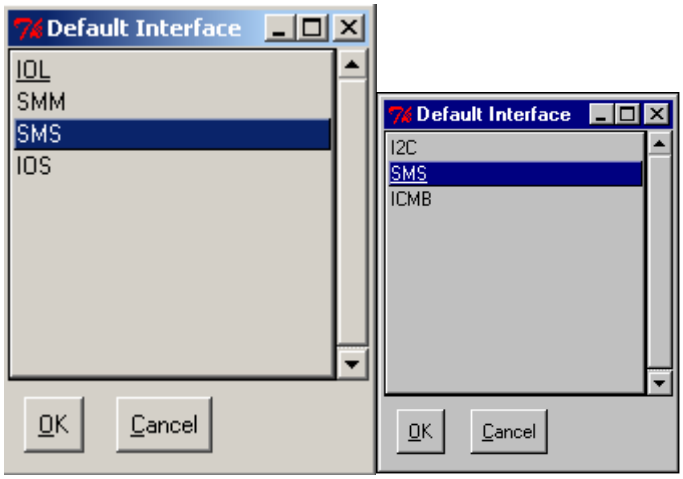


Figure 4-13. Default Interface Dialog Box

3. *Click* the interface item you want to set as the default.
4. *Click* OK to enable the new default interface. *Click* Cancel to return to the interface without change.

4.4 Saving and Retrieving User Settings

Once you have modified the framework to suit your needs and preferences, you can save the settings to the user configuration file. In addition to general appearance settings and the interface and transport settings, saving a user configuration file also saves the list of loaded tests. Invoking the framework with a saved user configuration file automatically loads the tests listed.

To save a user configuration file, complete the following procedure:

1. *Click* File in the menu bar at the top of the framework interface window. The File drop-down menu appears. Figure 4-14 shows the File drop-down menu:

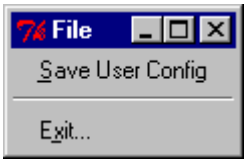


Figure 4-14. File Drop-down Menu

2. *Click* Save User Config. Your current settings are saved.

5 Running Conformance Tests

This chapter describes methods of running loaded test modules and analyzing the results. Before running a test module, the system must be installed and configured. For additional information on installing the framework and creating the necessary configuration files, refer to the "Installation and Configuration" chapter of this manual.

5.1 Running Tests

While it is only possible to execute one test at a time through the framework, that test may run other tests. A test that executes other tests is called a parent. The tests a parent runs are called child tests. The children may have children of their own. This hierarchy of parents and children is duplicated in the GUI menus described in the "User Interface" chapter of this manual. Running the test at the top of the hierarchy results in execution of all child tests, and thus the entire test suite.

Navigating the test hierarchy downward through the menus allows you to run any subset of the entire suite. At the bottom of the hierarchy you can run individual tests. Child tests are defined in the parent tests, and in the case of short versions of ICTS, children are hard-coded into the framework hierarchy. It is possible in FTF to write a parent test that determines its child tests on-the-fly, perhaps based on input from the user or a configuration file, but the non-developer's version of ICTS does not allow this.

To run a group of framework tests, complete the following procedure:

1. *Click* Test in the menu bar at the top of the framework user interface window. The Test drop-down menu appears. Figure 5-1 shows the Test drop-down menu:

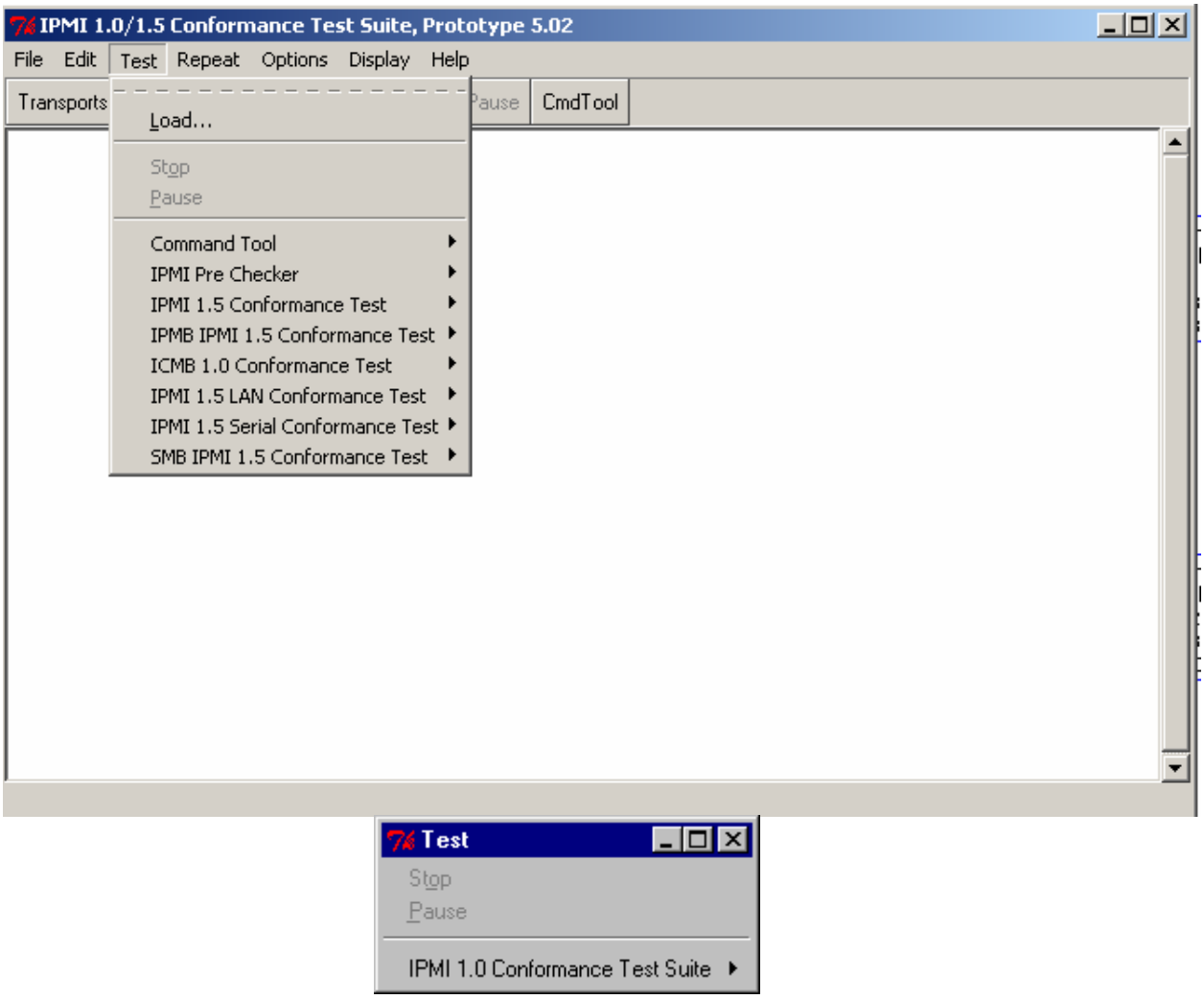


Figure 5-1. Test Drop-down Menu

2. Select IPMI 1.0/1.5 Conformance Test Suite. The IPMI Conformance Test Suite drop-down menu appears. Figure 5-2 shows the drop-down menu:

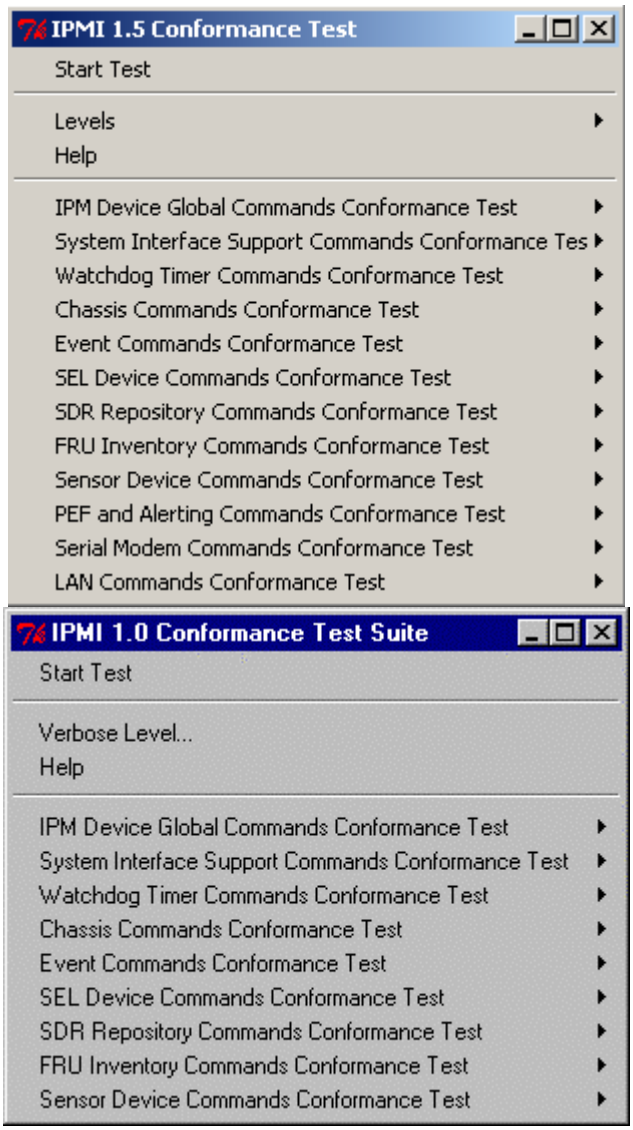


Figure 5-2. IPMI 1.0/1.5- Conformance Test Suite Drop-down Menu

3. Select the test group you want to run. A list of the tests in the group appears. Figure 5-3 shows the list of tests for the IPM Device Global Command Conformance Test:

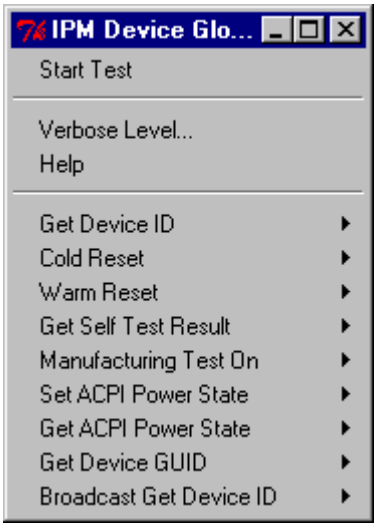


Figure 5-3. IPM Device Global Command Conformance Test Drop-down Menu

4. Select an individual test from the list. The Test menu for the selected test appears. Figure 5-4 shows the Test menu for the Get Device ID test:

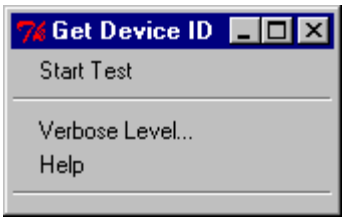


Figure 5-4. Get Device ID Dialog Box

5. Select Start Test... The framework begins the test and responses begin to appear in the framework message window.

NOTE

Selecting Start Test at any level runs the currently selected test and all its children.

5.2 Stopping or Pausing Tests

In many cases, it is possible to stop or pause a test in progress. However, not all tests support stops and pauses. Tests supporting safe stop and pause points make periodic API calls to the framework to seek the stop or pause request. Tests that do not make API calls cannot receive start and stop requests. This disables the framework's user interface for the duration of the test by preventing the Tcl/Tk interpreter from processing events in its event queue.

To stop a stop-enabled test, complete the following procedure:

1. From the Test drop-down menu, select Stop. There is no confirmation dialog box. A stop request is returned to the test, but the test must have the characteristics necessary to stop itself.

To pause a pause-enabled test, complete the following procedure:

1. From the Test drop-down menu, select Pause. A resume dialog box appears.
2. Resume the test or stop the test by clicking the appropriate button.

Index

G

H

I

K

L

M

O

P

R

S

T

U

V

W

/1.5