INTEGRITY Installation Guide

Green Hills Software

30 West Sola Street
Santa Barbara, California 93101
USA
Tel: 805-965-6044
Fax: 805-965-6343
www.ghs.com

DISCLAIMER

GREEN HILLS SOFTWARE MAKES NO REPRESENTATIONS OR WARRANTIES WITH RESPECT TO THE CONTENTS HEREOF AND SPECIFICALLY DISCLAIMS ANY IMPLIED WARRANTIES OF MERCHANTABILITY OR FITNESS FOR ANY PARTICULAR PURPOSE. Further, Green Hills Software reserves the right to revise this publication and to make changes from time to time in the content hereof without obligation of Green Hills Software to notify any person of such revision or changes.

Copyright © 1983-2015 by Green Hills Software. All rights reserved. No part of this publication may be reproduced, stored in a retrieval system, or transmitted, in any form or by any means, electronic, mechanical, photocopying, recording, or otherwise, without prior written permission from Green Hills Software.

Green Hills, the Green Hills logo, CodeBalance, GMART, GSTART, Slingshot, INTEGRITY, and MULTI are registered trademarks of Green Hills Software. AdaMULTI, Built With INTEGRITY, EventAnalyzer, G-Cover, GHnet, GHnetLite, Green Hills Probe, Integrate, ISIM, PathAnalyzer, Quick Start, ResourceAnalyzer, Safety Critical Products, SuperTrace Probe, TimeMachine, TotalDeveloper, velOSity, and μ-velOSity are trademarks of Green Hills Software.

All other company, product, or service names mentioned in this book may be trademarks or service marks of their respective owners.

Book compiled on February 28, 2018
RTOS Document Id: 54497

Contents

Chapter 1

Introduction

This chapter contains:

- The INTEGRITY Real-Time Operating System
- About This Manual
- INTEGRITY 11.7 Document Set

1.1 The INTEGRITY Real-Time Operating System

The INTEGRITY real-time operating system (RTOS) is a secure, high reliability system intended for use in mission-critical embedded systems. The INTEGRITY real-time operating system uses hardware memory protection to isolate and protect both itself and user tasks from incorrect operation caused by accidental errors or malicious tampering. Its object-oriented design allows verification of the security and integrity of data, communications, individual components, and the system as a whole. Its strict adherence to provable resource requirements allows an embedded system designer to guarantee resource availability.

Unlike other memory protected operating systems, the INTEGRITY real-time operating system does not sacrifice real-time performance for security and protection. It is first and foremost a real-time operating system.

1.2 About This Manual

The *INTEGRITY Installation Guide* provides information that is specific to installing and getting started with the INTEGRITY real-time operating system. Its purpose is to:

- Provide directions for installing the INTEGRITY binary distribution.
- Describe how to install optional items and licensed source code components.
- Explain how to connect to your target and run the INTEGRITY kernel.
- Describe the complete set of INTEGRITY Documentation.

This manual should be used in conjunction with *MULTI: Getting Started*, which provides installation instructions for the MULTI integrated development environment (IDE), compilers, and tools that are needed to build INTEGRITY projects.

This manual contains the following chapters:

- "INTEGRITY Software Installation" — describes how to install an INTEGRITY binary distribution from a CD.

- "Installing Encrypted and Other Optional Components" — discusses installation of encrypted libraries and source code, and other optional components into an existing INTEGRITY binary installation.

- "Getting Started with INTEGRITY" — walks you through the steps needed to boot INTEGRITY on the INTEGRITY simulator (ISIM) and download and run your first application.

Note: Previous versions of this book included the chapter, "Setting Up an x86 Target", which provides information about preparing an x86 or x64 PC to run INTEGRITY as its operating system. This chapter is now located in the *INTEGRITY GILA Boot Loader Guide*.

1.3 INTEGRITY 11.7 Document Set

INTEGRITY documentation is available in the following formats:

- Printed books (select books are not available in print).

- Online help, accessible from MULTI's **Help** menu.

  This method is generally the best option because the MULTI Help Viewer has powerful search capabilities and options for displaying online help. For more information see "The Help Viewer Window" in the "Introduction" chapter of *MULTI: Getting Started*.

- PDF documents for viewing in your favorite PDF viewer.

  PDF versions of documents are located in ***install_dir*\manuals**. This directory may also contain pdf-only manuals that are not included in the online help system or available in printed format.

The INTEGRITY 11.7 documentation set includes the following manuals:

- ***INTEGRITY Installation Guide*** — provides information that is specific to installing and using the INTEGRITY real-time operating system. The *Installation Guide* provides directions for installing the INTEGRITY binary distribution, describes how to install

optional items and licensed source code components, and explains how to connect to your target and run the INTEGRITY kernel.

- ***INTEGRITY Development Guide*** — describes the important files in an INTEGRITY installation and how to navigate the files to build system libraries (for standard source or kernel source installations) and user applications. The guide also describes how to perform run-mode application debugging as well as freeze-mode kernel debugging. The *Development Guide* is meant to be a central resource where the user can learn to maximize the benefits of the Green Hills MULTI Environment as it pertains to software development for INTEGRITY.

- ***INTEGRITY Kernel User's Guide*** — describes the major INTEGRITY RTOS concepts and provides the core INTEGRITY API description. The INTEGRITY API includes functions for manipulating all of the INTEGRITY Object types, including Tasks, Semaphores, Connections, MemoryRegions, IODevices, Activities, and AddressSpaces.

- ***INTEGRITY Kernel Reference Guide*** — contains reference material for the *INTEGRITY Kernel User's Guide*. It contains complete information about INTEGRITY kernel function calls, including a functional description, restrictions, arguments, and return values.

- ***INTEGRITY Libraries and Utilities User's Guide*** — contains information about a variety of system services and libraries, such as the INTEGRITY shell, INTEGRITY file system, POSIX (an alternate API to the standard INTEGRITY API), and device drivers.

- ***INTEGRITY Libraries and Utilities Reference Guide*** — contains reference material for the *Libraries and Utilities User's Guide*. This book contains function calls for a variety of system services libraries such as POSIX, file system, and device drivers. Functional description, restrictions, arguments, and return values are provided for each function.

- ***INTEGRITY Networking Guide*** — provides information about network configuration, network and sockets libraries, and the GHnet v2 TCP/IP stack, including the dual-mode IPv4/IPv6 stack. Also includes information about GHnet v2 optional services, such as the web server.

- ***INTEGRITY BSP User's Guide*** — describes the functions that must be defined and called when porting INTEGRITY to a new board. It also includes important information regarding how to write device drivers, install interrupt service routines, and other low-level software. A single ASP (Architecture Support Package) accommodates all the boards that are powered by the same processor family.

- ***Integrate User's Guide*** — contains information about using the Integrate utility. Integrate allows you to create multiple virtual AddressSpaces, each containing a variety of possible Objects such as Tasks, Semaphores, and Connections. INTEGRITY supports the common

method of creating such objects dynamically via kernel calls, with Integrate you have the option of statically specifying a variety of RTOS objects and relationships, saving code development time.

- ***EventAnalyzer User's Guide*** — provides information about using the MULTI EventAnalyzer to collect and analyze data. The EventAnalyzer helps programmers understand the complex interaction of resources in a real-time operating system by providing a graphical representation of system activities. With the EventAnalyzer, system events occurring within microseconds can easily be isolated and analyzed. This system analysis tool effectively gives programmers the ability to stop time in order to scrutinize system behavior.

- ***INTEGRITY Implementation-Dependent Behavior Guide*** — provides information about implementation-dependent behavior. Because implementation-dependent behavior can change between minor INTEGRITY releases, you should use this manual for general guidelines, but be aware that details may change without notice.

1.3.1 MULTI Document Set

In addition to the INTEGRITY document set, the Green Hills toolchain comes with a comprehensive set of documentation to aid developers using Green Hills compilers, toolchain, and the MULTI Integrated Development Environment (IDE). These books are available as printed manuals, as **pdf** files, and through MULTI's **Help** menu.

INTEGRITY manuals focus on the tools that an INTEGRITY user is most likely to use and often refer to the MULTI manuals. Programmers (especially those who are just starting to use MULTI for the first time) will find the MULTI manuals to be useful companions. While the INTEGRITY manuals focus on high level tools and commonly used commands, the MULTI manuals provide detailed information about each tool. Some of the information in MULTI manuals is not pertinent to INTEGRITY users, or superseded by methods described in INTEGRITY manuals.

The primary documents in the MULTI document set are:

- ***MULTI: Getting Started*** — provides an introduction to the MULTI Integrated Development Environment (IDE) and leads you through a simple tutorial.
- ***MULTI: Licensing*** — describes how to obtain, install, and administer Green Hills licenses.
- ***MULTI: Managing Projects and Configuring the IDE*** — describes how to create and manage projects and how to configure the MULTI IDE.
- ***MULTI: Building Applications*** — describes how to use the compiler driver and the tools that compile, assemble, and link your code. Also describes the Green Hills implementation of supported high-level languages. Some of the information in *MULTI: Building Applications* is not pertinent to INTEGRITY users, or superseded by information in INTEGRITY manuals. For example:
  - The command-line driver descriptions are geared towards building applications for stand-alone use. Although many of the processor-specific driver options are supported

on INTEGRITY systems, this information is superseded by the information in the "Using the Command-line Driver" section of the *INTEGRITY Development Guide*.

- Information on reentrancy of runtime libraries in the "Libraries and Header Files" chapter is augmented by the "Thread-Safety of Run-time Libraries" section of the *INTEGRITY Libraries and Utilities User's Guide*.
- The "Customizing the Green Hills Run-Time Environment" section of the "Libraries and Header Files" chapter is not applicable to INTEGRITY users because INTEGRITY provides a full-featured run-time environment out of the box for developers.
- Some of the special embedded systems features described in *MULTI: Building Applications*, such as position independent code, are not pertinent or supported on INTEGRITY systems.

- ***MULTI: Configuring Connections*** — describes how to configure connections to your target.
- ***MULTI: Debugging*** — describes how to set up your target debugging interface for use with MULTI and how to use the MULTI Debugger and associated tools.
- ***MULTI: Debugging Command Reference*** — explains how to use Debugger commands and provides a comprehensive reference of Debugger commands.
- ***MULTI: Scripting*** — describes how to create MULTI scripts. Also contains information about the MULTI-Python integration.

1.3.2 Conventions Used in Green Hills Documentation

Green Hills documentation uses the following typographical conventions to present information.

Bold Type indicates:

- Filenames or pathnames
- Commands
- Options
- Window titles
- Program names
- Button names
- Menu names or menu items
- Field names

Italic Type indicates:

- Titles of books
- Text that the user should replace with an appropriate argument, command, filename, or other value
- New terms or emphasis

`Monospace Font` indicates:

- Code samples
- Text that should be entered exactly as presented

1.3.3 Useful Links

- Green Hills Software website — www.ghs.com
- Green Hills Support — https://support.ghs.com
- Green Hills Support email — support@ghs.com

Chapter 2

INTEGRITY Software Installation

The INTEGRITY installation CDs contain the source code, precompiled libraries and related files needed to build and use the INTEGRITY RTOS. The Green Hills MULTI Integrated Development Environment (IDE) and related compilers and tools must be installed separately.

This chapter covers the following topics:

- Installation Restrictions
- Running the INTEGRITY Installer
- Installing MULTI IDE and Compiler Patches
- TFTP Server Setup (UNIX Only)
- Evaluation Versions
- Uninstalling INTEGRITY

The INTEGRITY RTOS has been ported to a variety of common boards. If you have one of these boards set up and available, you can proceed with development immediately after completing the software installation. To develop code and run INTEGRITY on a board that does not have a Green Hills-supplied reference port, a Board Support Package (BSP) must first be developed for the board. For more information, see the *INTEGRITY BSP User's Guide*.

For additional information, see the "INTEGRITY Installation and Directory Structure" chapter in the *INTEGRITY Development Guide*.

2.1 Installation Restrictions

Use a separate directory for each installation:

- Do not install Green Hills tools on top of an INTEGRITY installation.
- Do not install INTEGRITY on top of existing Green Hills tools.
- Do not install this INTEGRITY RTOS on top of other INTEGRITY installations unless the version numbers are exactly the same and you are adding support for additional architectures.

- Do not change the toolchain being used to build any specific installation without rebuilding every installed BSP in the installation with the new toolchain.

Note: The order in which you install MULTI and INTEGRITY does not matter, and you may have several versions of each installed. However, we recommend installing MULTI before installing INTEGRITY.

2.1.1 System Requirements

For current information about system requirements and MULTI version requirements, see the "Installation" section of the *INTEGRITY Release Notes*.

2.2 Running the INTEGRITY Installer

The INTEGRITY CD contains an installer program for each supported host operating system. The user performing the installation must have permission to write to the necessary directories and must be able to create directories that are accessible to other INTEGRITY users.

- On Windows — the installing user must be able to write to the Windows System directories. On most Windows systems the installing user must be a member of the Administrators group. The installing user should ensure that end users will have full access to the installed files.

- On Linux and UNIX — the installing user should set the "group" and "other" permissions of installation files to appropriate values according to your company's IT policies. It is not necessary or recommended to install from the "root" user account, only that the installing user be able to write to the installation directory.

The INTEGRITY installer contains several screens that are described in the following instructions. Some screens are only applicable to certain INTEGRITY installations. The installer automatically hides screens that do not apply to your distribution or that require license keys you have not provided.

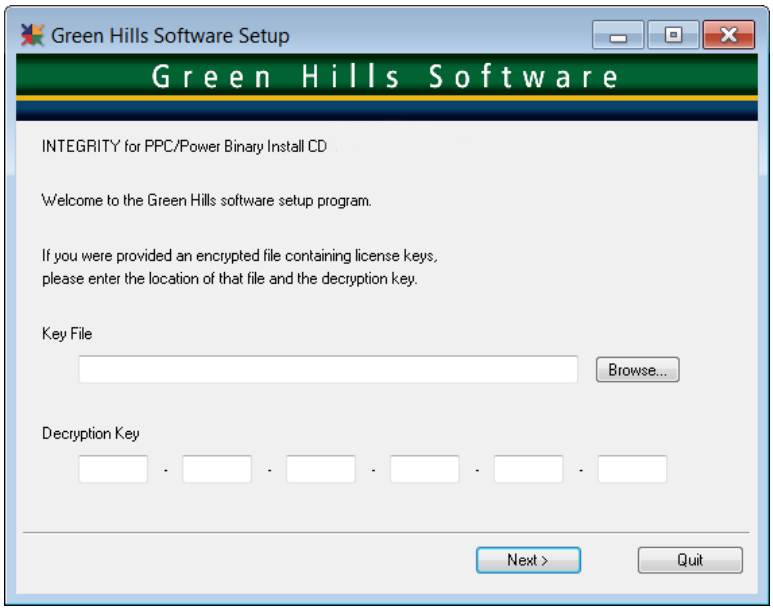
To install INTEGRITY:

1. Execute the appropriate program for your host operating system. The following installer programs are provided:

   - **ginstall_win32** — for Windows
   - **ginstall_solaris2** — for Sun Solaris
   - **ginstall_linux86** — for Linux (on x86 platforms)

   Note: To install INTEGRITY as the operating system on an x86 or x64 target, you must first set up the x86 or x64 PC using the instructions in the "Setting Up an x86 Target" chapter of the *INTEGRITY GILA Boot Loader Guide*.

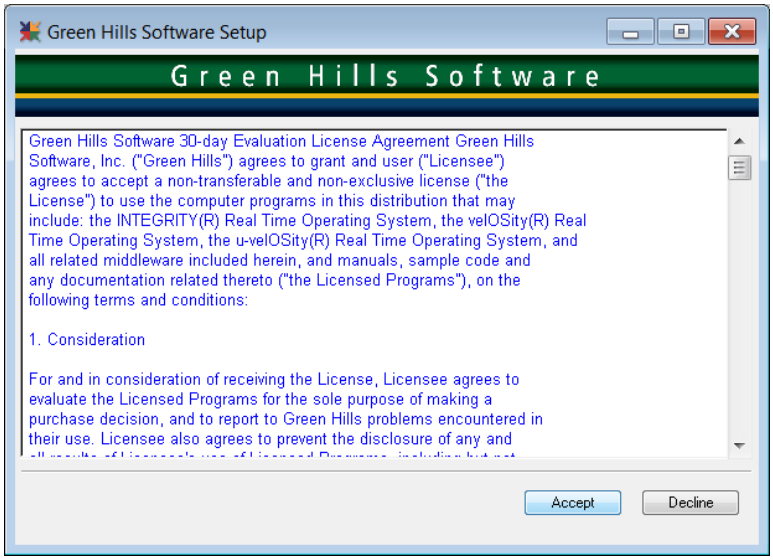
2. The first screen of the installer indicates the version of the software being installed and provides fields for entering a License **Key File** and **Decryption Key**.



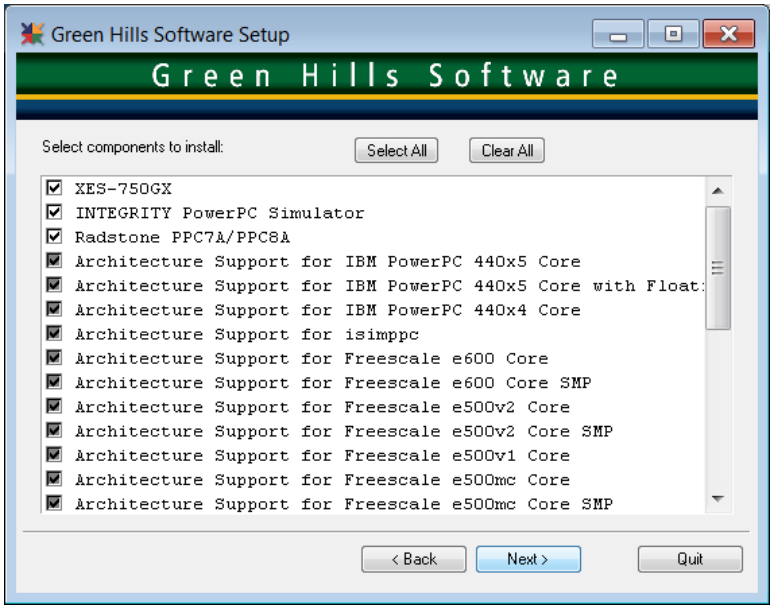
- If you have received a License Key File, use this screen to browse or enter the filename, then enter the decryption key.

  Note: Decryption keys must be kept secure and it is the customer's responsibility to protect keys to source code. Do not share decryption keys with anyone and do not transmit decryption keys via unsecured channels, such as email.

- If you do not have such a file, click **Next** to open the Evaluation License screen, which allows installation of INTEGRITY for purposes of product evaluation only if you agree to the license terms displayed. This screen will not display if a valid decryption key was provided. Click **Accept** if you agree to the license terms.



3. On the next screen, select the check boxes for the components you want to install. Encrypted components will only display in this list if you have previously entered a valid key file and decryption key. The INTEGRITY Simulator, if available, is selected for installation by default.



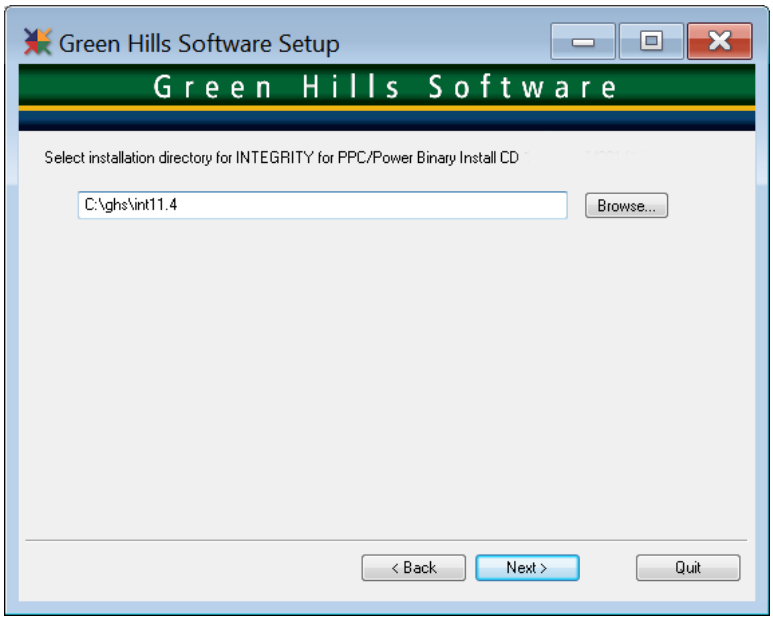
Note: You can run the installer again later to install additional BSPs. For information, see "Adding Components to an Existing Installation" in the "Installing Encrypted and Other Optional Components" chapter.

In addition to installing specific BSPs, you can select generic support for a particular architecture, which installs the INTEGRITY kernel and other libraries for that architecture, even if a BSP using that architecture is not selected for installation. For example, a PowerPC installation might provide the following generic support packages:

- Architecture Support for Freescale e500v1 Core
- Architecture Support for Freescale e500v2 Core
- Architecture Support for Freescale e500v2 Core SMP
- Architecture Support for Freescale e500mc Core
- Architecture Support for Freescale e500mc Core SMP
- Architecture Support for IBM PowerPC 440x4 Core
- Architecture Support for IBM PowerPC 440x5 Core
- Architecture Support for IBM PowerPC 440x5 Core with Floating Point

These generic support packages can be useful when writing your own BSP. If you are only using standard BSPs included in the distribution, you do not need to select any generic architecture support packages. The installer automatically selects and installs generic architecture support needed for standard BSPs installed.

4. On the next screen, specify the directory for this INTEGRITY installation. The installation directory should not refer to a previous INTEGRITY installation except when adding optional components to an existing installation. Each version of INTEGRITY should be installed into a separate directory.

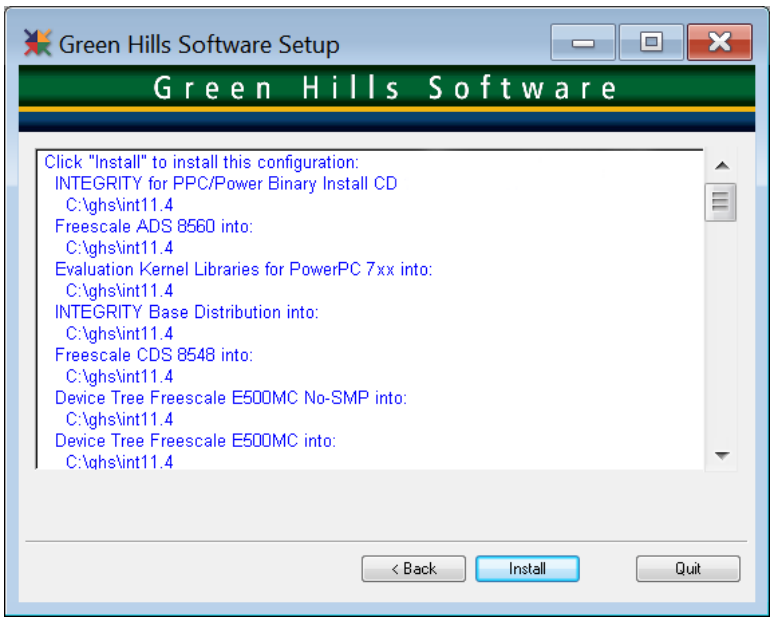


The default installation directories for INTEGRITY are:

- Windows — **C:\ghs\int*version***
- Linux — **/usr/ghs/int*version***

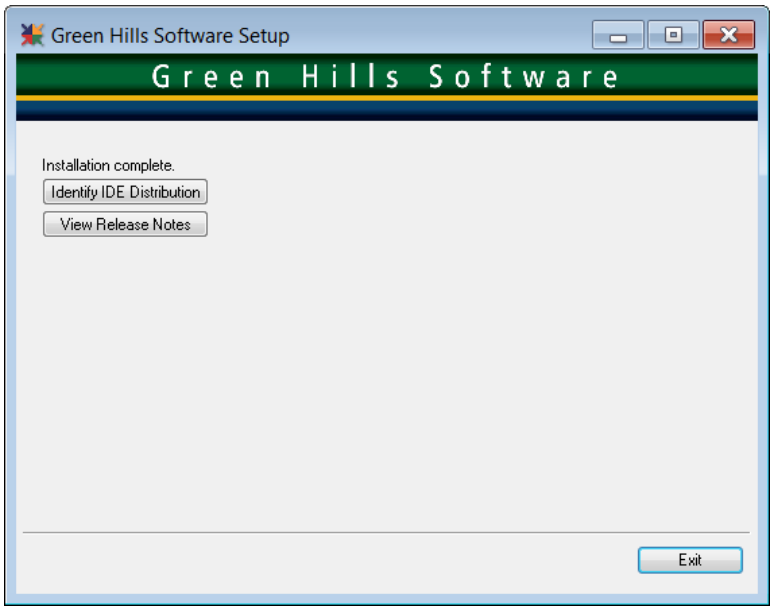
where ***version*** is replaced by the INTEGRITY version number with decimal points removed (for example, **int1170** for INTEGRITY 11.7.0).

5. The next screen displays a summary of the installation and options selected in the previous screens. Click **Install** to begin the installation process.



Throughout the installation process, the screen will be updated with the steps currently being performed.

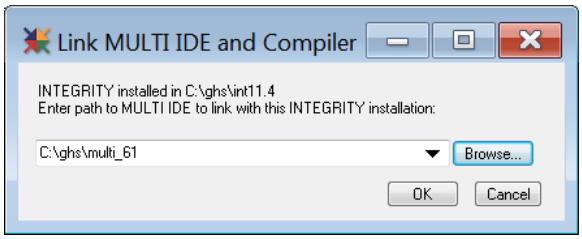
6. When the installation process has completed, the Installation Complete screen will display.



You can use the buttons on this screen to do the following:

(a) To configure your MULTI installation to work with INTEGRITY, you can specify the location of the MULTI IDE and Compiler. Click the **Identify IDE Distribution** button on the Installation complete screen, to open the following dialog:

Note: This is an interface to the **integrity.dist** file in MULTI's **config** directory. If you have previously specified your INTEGRITY distribution in MULTI, the user-specific setting will override any settings made in this dialog and you should skip this step. For information about setting the user-specific **integrity.dist** file, see "Configuring MULTI for Use with INTEGRITY or u-velOSity" in the *MULTI: Getting Started* manual.



Select the correct IDE installation directory from the drop-down, enter the pathname, or browse to navigate to the correct directory, then click **OK**.

(b) To open the INTEGRITY release notes, click the **View Release Notes** button. The release notes will open, provided the installer is able to identify an appropriate PDF viewer on your system. Release notes are also installed into the **manuals** directory of the installation for viewing offline. You may wish to contact support for the latest version of the release notes, as new information may have been added after product release.

7. Click **Exit** to close the installer when done.

Note: After installing INTEGRITY, you must build **default.gpj** for your BSPs and architectures in the MULTI Project Manager.

2.3 Installing MULTI IDE and Compiler Patches

Some INTEGRITY installations will require that you install MULTI IDE and/or Compiler patches. See the "Required MULTI IDE and Compiler Patches" section of the *INTEGRITY Release Notes* for details about patches that apply to your INTEGRITY installation.

To install a patch, use the **gpatch** utility located in ***rtos_install_dir*/multi/bin/*host*** (or a newer version of **gpatch**, if available) as follows:

1. From a command prompt, navigate to the install directory for the MULTI IDE or Compiler that is going to be patched.

2. Use **gpatch** as follows:

   `gpatch` *`install_CD_path`*`\patch_`*`number_host`*`.iff`

   - *`install_CD_path`* specifies the full path to the location of the patch on the INTEGRITY installation CD.
   - *`number`* specifies the patch number to be installed.
   - *`host`* specifies the host on which you are installing and will be one of `solaris2`, `win32`, or `linux86`. Make sure you install the correct patch for your host.

Note: After installing any patches, you must rebuild **default.gpj** for your BSPs in the MULTI Project Manager.

2.4 TFTP Server Setup (UNIX Only)

A TFTP server is required when using onboard firmware that uses TFTP to network boot INTEGRITY kernels or other applications.

To enable a TFTP Server for UNIX:

1. In the **/etc/inetd.conf** file, uncomment the `tftp` line.

2. It may also be necessary to remove the **-s** command in the **in.tftpd** command.

3. After modifying the file, reboot your system or restart the **inetd** daemon (via `kill -HUP PID` where `PID` is the process ID of the inetd server).

   Some Linux host systems may use **xinetd** instead of the traditional UNIX **inetd**. For **xinetd**, edit the file **/etc/xinetd.d/tftp** and set **Disable** to **No**. You may still need to send the HUP signal to the xinetd process as above.

If you are having problems starting the TFTP server, consult your system administrator for help.

2.5 Evaluation Versions

Evaluation versions of the kernel and optional components contain timebomb code. These components stop functioning after the duration of a certain amount of time (usually two hours), or after a certain number of operations. To eliminate the timebomb for components, install the source code or a licensed version of the component using a valid decryption key, and rebuild as discussed in the "Installing Encrypted and Other Optional Components" chapter.

For customers with an object code-only license to the INTEGRITY kernel, pre-built licensed kernel libraries are encrypted on the binary installation CD. These licensed kernel libraries should appear as optional components in the encrypted components screen (see "Running the INTEGRITY Installer").

Note: Kernel libraries are often shared among many targets, so they are typically provided for each architecture family, as with the system and C/C++ runtime libraries. In cases where the libraries are built in a BSP-specific way, the libraries are provided on a per-BSP basis (for example, on the INTEGRITY simulators). If you are unsure about which kernel library packages are appropriate, there is no harm in installing all of them.

2.6 Uninstalling INTEGRITY

To uninstall INTEGRITY on Windows:

- Remove the program using the Control Panel.

  Note: Uninstalling INTEGRITY on Windows will not remove new or modified files from the installation directory. You can remove these items manually.

To uninstall INTEGRITY on a Linux system:

- Remove the installation directory. For example, to remove an existing INTEGRITY installation in **/export/ghs/int1100/**, use the following command:

  ```
  rm -Rf /export/ghs/int1100
  ```

  Note: Removing a Linux installation will also remove any new or modified files and projects you may have within the distribution.

Chapter 3

Installing Encrypted and Other Optional Components

This chapter provides information about installing encrypted or unencrypted components that are not part of a standard binary installation. This chapter covers the following topics:

- CD Contents
- Adding Components to an Existing Installation

Note: Decryption keys are required to install INTEGRITY source code and licensed libraries. If you are not licensed to install INTEGRITY source code or licensed libraries, you can build INTEGRITY applications by linking against prebuilt evaulation libraries.

3.1 CD Contents

This section provides information about the content of installation CDs.

Source code and licensed libraries for optional components ships in encrypted form unless otherwise noted.

Many libraries are included on both Binary and Standard Source CDs. When a library is included on both, the Binary CD only contains the prebuilt libraries, and does not contain the source for these libraries. The Standard Source CD contains source for most licensed libraries.

In some cases, libraries or source packages depend on other libraries or source packages also being installed. If you install a library or source package with a dependency, you must ensure that any package listed as a prerequisite is also installed before attempting to rebuild your BSP.

If you have received decryption keys for any licensed libraries or source code, or want to use any of the unencrypted components, you can install the source code using the instructions in the "Adding Components to an Existing Installation" section.

Note: Decryption keys must be kept secure and it is the customer's responsibility to protect keys to source code. Do not share decryption keys with anyone and do not transmit decryption keys via unsecured channels, such as email.

3.1.1 Binary CD Components

A Binary CD contains the following components:

- INTEGRITY BSP Source
- Source code for special BSP-specific libraries
- Licensed INTEGRITY Kernel Libraries for *board or architecture*
- Licensed INTEGRITY Middleware Libraries
- Licensed Atheros AR6000 Driver Libraries
- Licensed Network Server Libraries

The BSP Source package on a binary CD contains BSP source for general BSP support components (e.g., PCI support) and source corresponding to the BSPs on the binary CD. If you are using multiple binary CDs (e.g., a PowerPC binary CD and an ARM binary CD), you should install the BSP Source packages on each of the installation CDs.

BSP Source for BSPs based on QorIQ processors with Data Path Acceleration Architecture (DPAA) requires installation of the Freescale NetComm software sources, which are included on binary CDs containing these BSPs. Failure to install the NetComm software package may result in build failures. If this occurs, re-run the installer to install the NetComm software package.

3.1.2 Standard Source CD Components

A Standard Source CD contains encrypted source files for INTEGRITY libraries and components, with the exception of the following:

- BSP Source — provided on Binary CD.
- Kernel Source — provided on Kernel Source CD.
- Atheros AR6000 Driver Library — binary only, provided on Binary CD.
- Network Server Source — part of BSP Source package, provided on Binary CD.

3.1.3 Kernel Source CD Components

A Kernel Source CD contains the following components:

- INTEGRITY System Source Package

The INTEGRITY System Source package requires that the BSP source packages for all BSPs have already been installed. Installing System Source without the corresponding BSP Source packages may result in build failures. If this occurs, re-run the installers for each binary CD to install the appropriate BSP source package(s).

3.2 Adding Components to an Existing Installation

If you have previously installed a version of INTEGRITY and you wish to install components you did not select at the time of the original installation, you may do so at any time. However, after installing components, you must rebuild all installed BSPs as described in "Getting Started with INTEGRITY". Please read the remainder of this chapter before installing additional components for any special directions or considerations regarding the component you wish to install.

Note: Installing BSPs after your initial installation may cause previously installed licensed kernel libraries to be replaced with the timebombed libraries. If this occurs, you will need to reinstall the licensed kernel libraries.

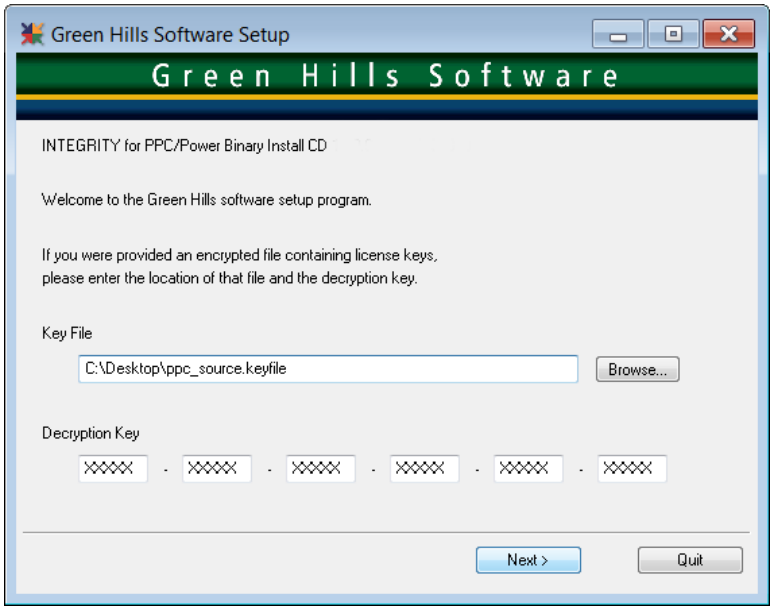
1. To install most components, use the CD that contains the component you want to install.

2. Execute the appropriate installer program as described in "Running the INTEGRITY Installer" in the "INTEGRITY Software Installation" chapter.

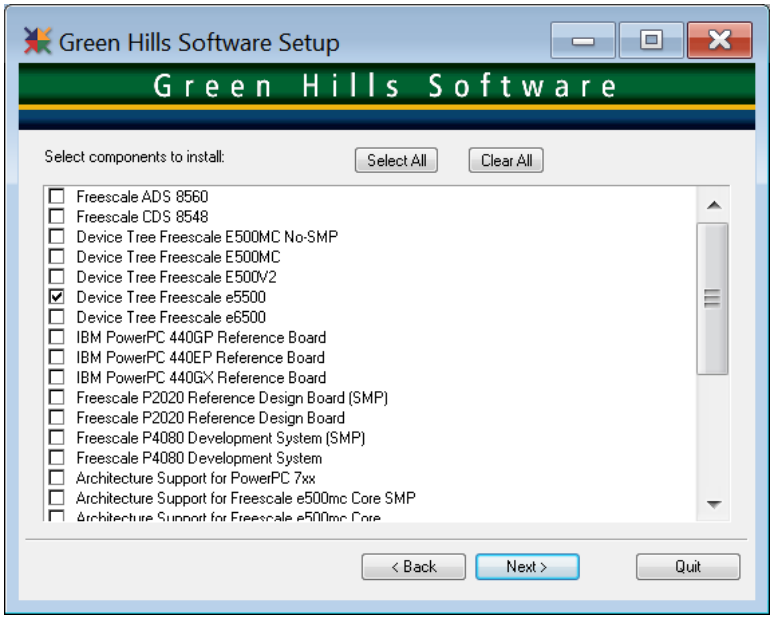
   - **ginstall_win32** — for Windows
   - **ginstall_linux86** — for Linux (on x86 platforms)
   - **ginstall_solaris2** — for Sun Solaris

   The installer will proceed through the steps described in the "Running the INTEGRITY Installer" section.

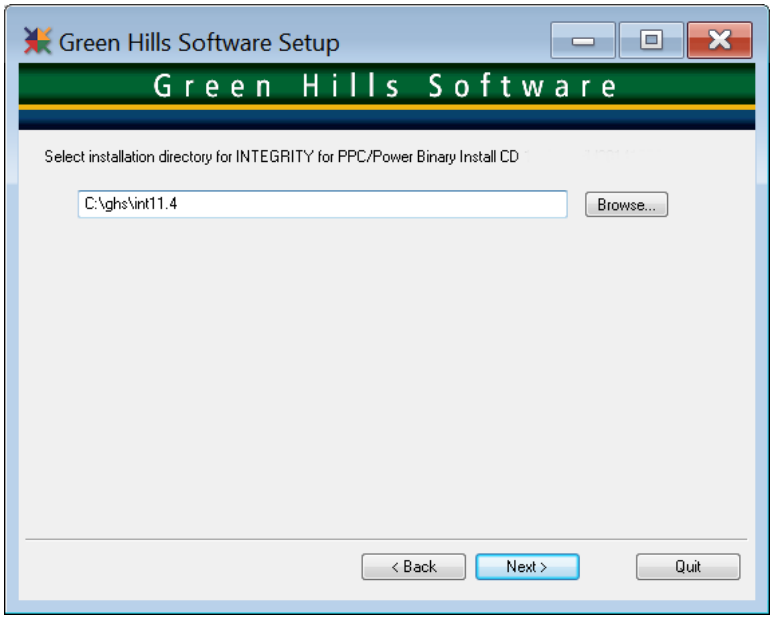
3. The Welcome screen of the installer will open. Use this screen enter a License **Key File** and **Decryption Key**.



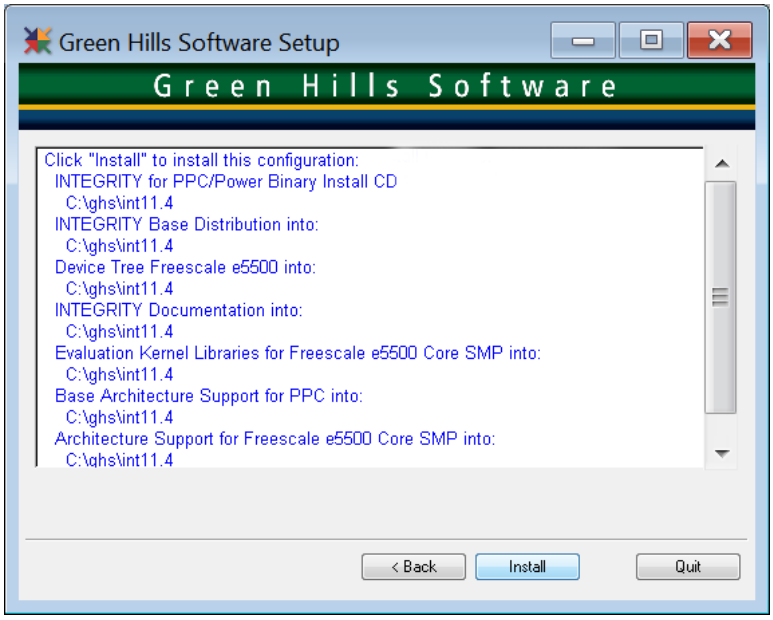
4. Select the BSPs or optional components you wish to add.



5. On the next screen, select the location of your existing INTEGRITY installation.



6. Proceed to the Summary screen and click **Install** to install the selected items.



Note: After adding any components to an existing installation, you must rebuild **default.gpj** for your BSPs with the MULTI Project Manager.

Chapter 4

Getting Started with INTEGRITY

After INTEGRITY and the MULTI Integrated Development Environment have been successfully installed, every installed BSP must be built before new projects can be created or a kernel can be booted. In addition, if new BSPs or source packages are added to the INTEGRITY installation at a later time, or the MULTI tools used with the INTEGRITY distribution are modified, all installed BSPs must be built again to ensure that all libraries are up-to-date, evaluation time limits are removed, and shared libraries are linked with the appropriate run-time libraries provided with the toolchain. This chapter focuses on using the INTEGRITY Simulator to get started, but the basic steps for getting started also apply to hardware targets.

This chapter contains the following sections:

- Starting the MULTI Launcher
- Creating MULTI Workspaces for Installed BSPs
- Building Installed BSPs
- Running the Kernel in ISIM and Connecting with rtserv2
- Target Board Setup
- Booting an INTEGRITY Kernel

This chapter contains step-by-step instructions for downloading a kernel using the INTEGRITY Simulator (ISIM) as the target. ISIM is an ideal method to get up and running quickly because there are so few contingencies. The ISIM instructions provide a good example because the process of booting an INTEGRITY kernel is very simple and will be the same for every user. This process is summarized in the "Running the Kernel in ISIM and Connecting with rtserv2" section of this chapter, and described in greater detail in the "ISIM - INTEGRITY Simulator" chapter of the *INTEGRITY Development Guide*.

After you have connected to ISIM you can go through tutorials in the "Using INTEGRITY Tutorials" section of the *INTEGRITY Development Guide*, The tutorials provide step-by-step instructions for building new applications and running example applications that are included in the installation. They are an excellent way to quickly understand how to develop applications with the INTEGRITY RTOS and the MULTI Integrated Development Environment (IDE).

After the tutorials have been mastered, the **examples** directory contains additional INTEGRITY applications that demonstrate commonly used aspects of the INTEGRITY API and other layered software components included with the installation. For more information about how to use these examples, see "Copying Examples with the MULTI Project Manager" in the "Configuring Applications with the MULTI Project Manager" chapter of the *INTEGRITY Development Guide*.

Connecting to an actual board requires additional configuration steps not covered in the ISIM instructions. Some of these issues are described in the "Target Board Setup" and "Booting an INTEGRITY Kernel" sections later in this chapter. Additional information is in the **.notes** file for each BSP.

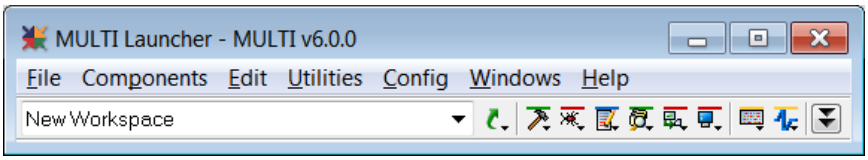
4.1 Starting the MULTI Launcher

The MULTI Launcher provides a convenient way to launch frequently used tools, create new files and projects, access recently used files and projects, and manage workspaces. The Launcher's **Windows** menu also provides a centralized window manager you can use to access any of your open MULTI windows.

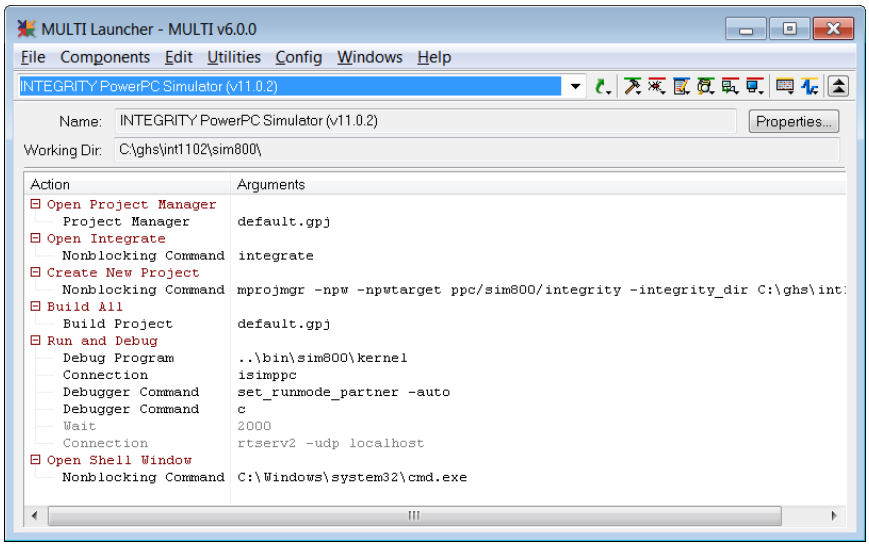
To start the MULTI Launcher:

1. Select **Start⇒Programs⇒Green Hills Software⇒MULTI *release name*⇒MULTI**.

   The MULTI Launcher will open.



2. If the detail pane is hidden, click the **Show Detail Pane** () button.



The main MULTI components can be accessed with buttons on the Launcher toolbar:

 — Runs a shortcut or an action sequence in the current workspace. Also allows you to create a new workspace or create or edit a shortcut.

 — Opens the Project Manager on a recent or new project.

 — Opens the Debugger on a recent or new executable.

 — Opens the Editor on a recent or new file.

 — Opens the Checkout Browser on a recent or new checkout.

 — Opens the Connection Organizer or a recent or new target connection.

 — Opens a Serial Terminal using a recent or new connection.

 — Opens the EventAnalyzer (licensed separately).

 — Opens the ResourceAnalyzer (licensed separately).

 — Closes the MULTI Launcher (UNIX only by default).

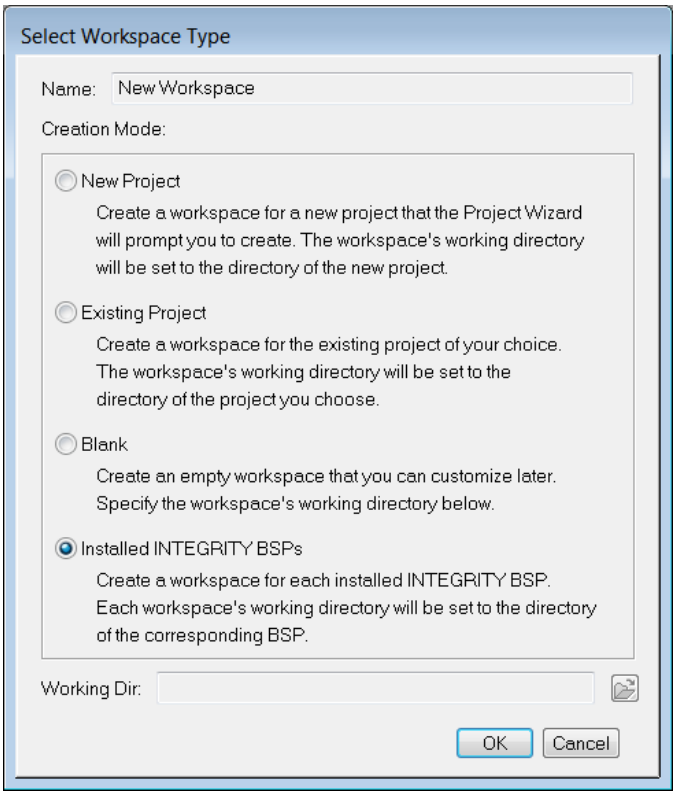
 — Shows/hides the detail pane of the Launcher.

4.2 Creating MULTI Workspaces for Installed BSPs

The MULTI Launcher can automatically create default workspaces for BSPs you have installed. These workspaces add shortcuts to the Launcher that perform common tasks such as opening the Project Manager, building all INTEGRITY source code for the BSP, creating new projects, and opening Integrate. For simulators, MULTI creates Launcher shortcuts that automatically launch an INTEGRITY simulator (ISIM) and run an INTEGRITY kernel within it. This section provides instructions for creating these default workspaces. After you have created a workspace, you can modify it to suit your needs, or create additional workspaces as needed.

For more information about using the MULTI Launcher and creating workspaces, see the "Managing Workspaces and Shortcuts with the Launcher" chapter of *MULTI: Managing Projects and Configuring the IDE*.

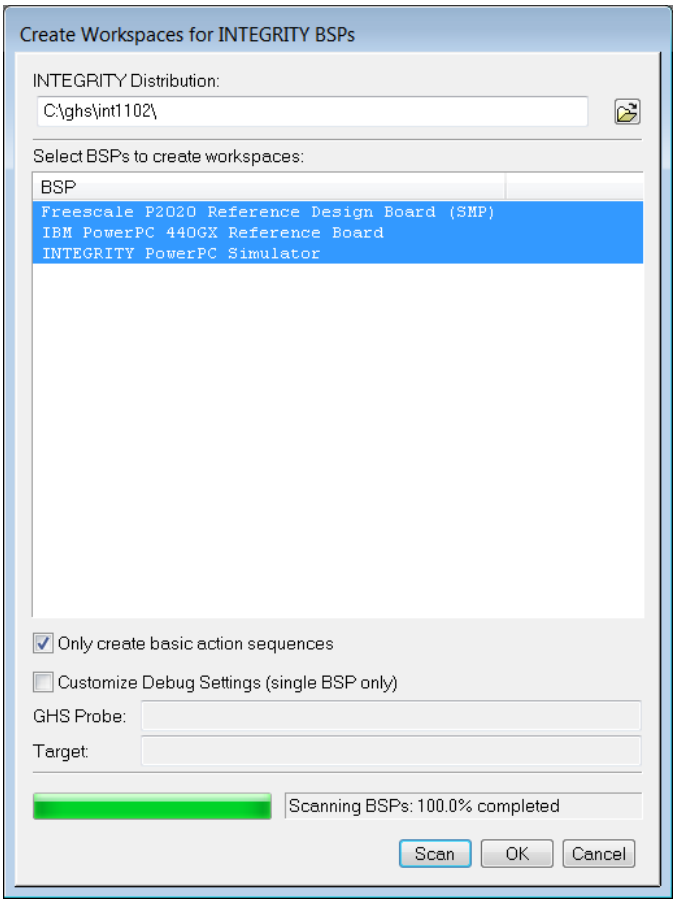
1. From the MULTI Launcher, select **File⇒Create Workspace**. The Select Workspace Type dialog will open.



2. Select the **Installed INTEGRITY BSPs** radio button and click **OK**.

3. Enter the location of your INTEGRITY installation in the **INTEGRITY Distribution** text box (if it is not already set correctly) and click **OK**. The MULTI Launcher scans the directory and displays all detected BSPs.

4. Highlight all of the BSPs for which workspaces should be created. Be sure to highlight the INTEGRITY simulator BSP you will be using for the remainder of this section. When finished, click **OK**.

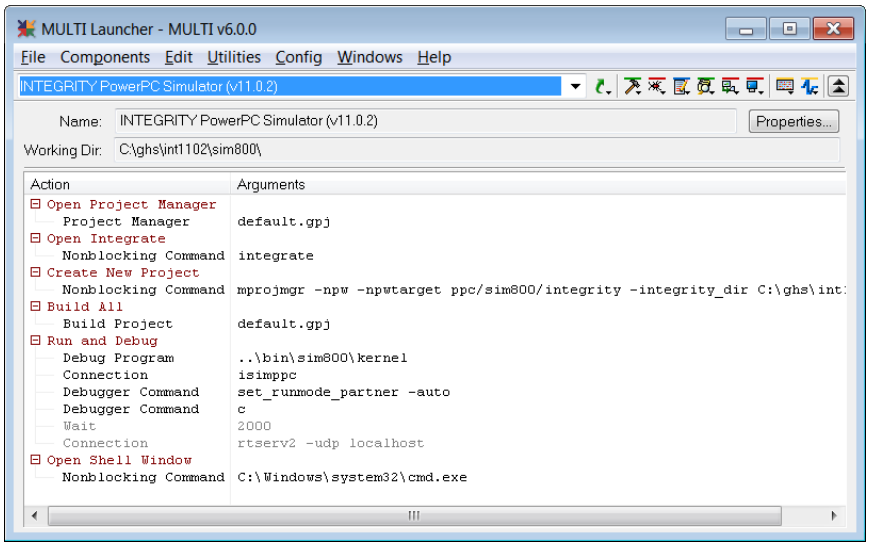


The MULTI Launcher creates default workspaces for each BSP you selected. The workspace names are derived from the name of the board. For example, the INTEGRITY PowerPC simulator workspace is named **INTEGRITY PowerPC Simulator**.

4.3 Building Installed BSPs

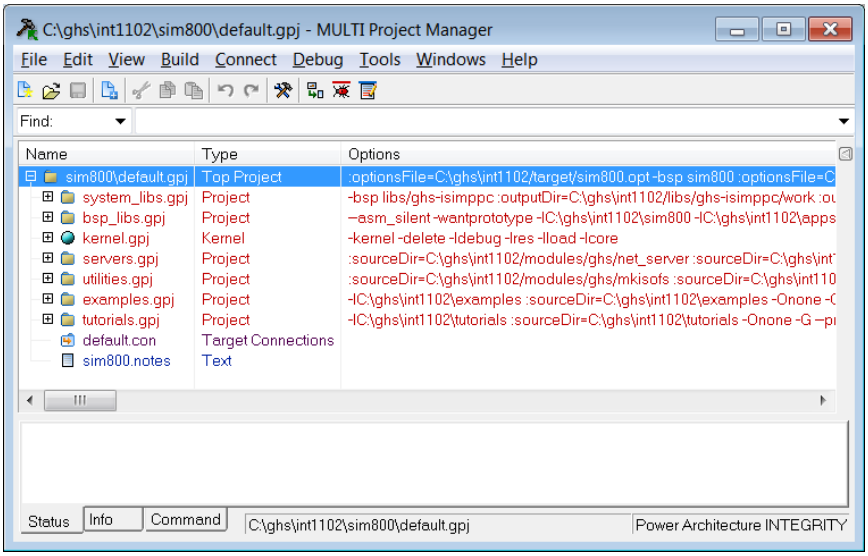
After installing INTEGRITY, you must rebuild all installed BSPs before creating new projects or booting a kernel. The instructions in this section use the Simulator BSP, but these same steps must be followed for all installed BSPs after installation, and anytime you install a new version of MULTI tools or add components to your installation. Failing to rebuild a BSP can result in later project build failures (missing libraries), inconsistent behavior between static and shared libraries, and other problems. Building the BSP is also a prerequisite for performing the tutorials in the "Using INTEGRITY Tutorials" section of the *INTEGRITY Development Guide*. If you are new to using MULTI to create INTEGRITY applications, we recommend that you perform the tutorials.

If the MULTI Launcher is not already open, use the instructions in "Starting the MULTI Launcher" to open MULTI.

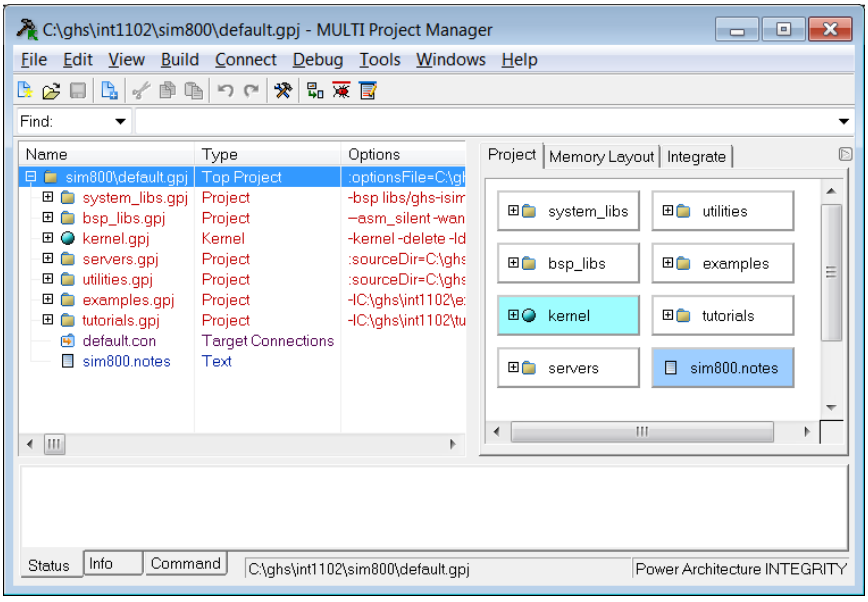


1. Use the Workspace selection drop-down menu to select the BSP you want to build.

2. From the Launcher, double-click the **Open Project Manager** action to open **default.gpj**. The MULTI Project Manager window will open with the default view as follows:



3. Select **View⇒Show All Views** or click the **Show View Tabs** arrow on the right border to display Advanced Views tabs in the right pane.



4. If you have installed new MULTI tools or INTEGRITY components, you must clean the build of any installed BSPs before proceeding. To do this, select **Build⇒Clean default.gpj**.

5. Select **Build⇒Build Top Project default.gpj**. This builds **default.gpj** for the BSP, including the examples and tutorials for the simulator.

6. Repeat this process for all installed BSPs.

4.4 Running the Kernel in ISIM and Connecting with rtserv2

This section describes the steps needed to boot an INTEGRITY kernel in the INTEGRITY Simulator (ISIM) and connect to the running kernel with the run-mode debugger, **rtserv2**.

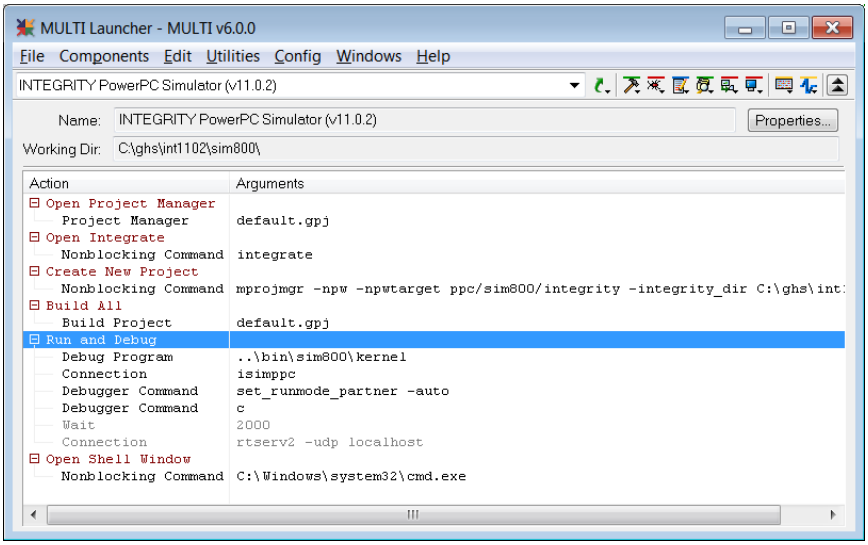
ISIM is a simulation environment for the INTEGRITY real-time operating system. It enables programmers to develop and test embedded INTEGRITY-based applications when target hardware is not yet manufactured, or in supply too limited to accommodate the entire programming team. Unlike conventional RTOS simulators that run as native UNIX or PC applications, ISIM simulates the same code that runs on the target. Thus, application size characteristics are known during simulation. In addition, the exact same compilers and development tools are used for both simulation and for actual hardware development. This increases the value of the development tools investment and the productivity of the programming team. ISIM provides complete INTEGRITY simulation, including virtual memory and memory protection. If you are unfamiliar with the MULTI IDE, we recommend that you begin by using the INTEGRITY Simulator (ISIM) as your target.

The MULTI Launcher adds a very powerful action sequence into INTEGRITY simulator workspaces: **Run and Debug**. This action sequence automatically performs the following steps:

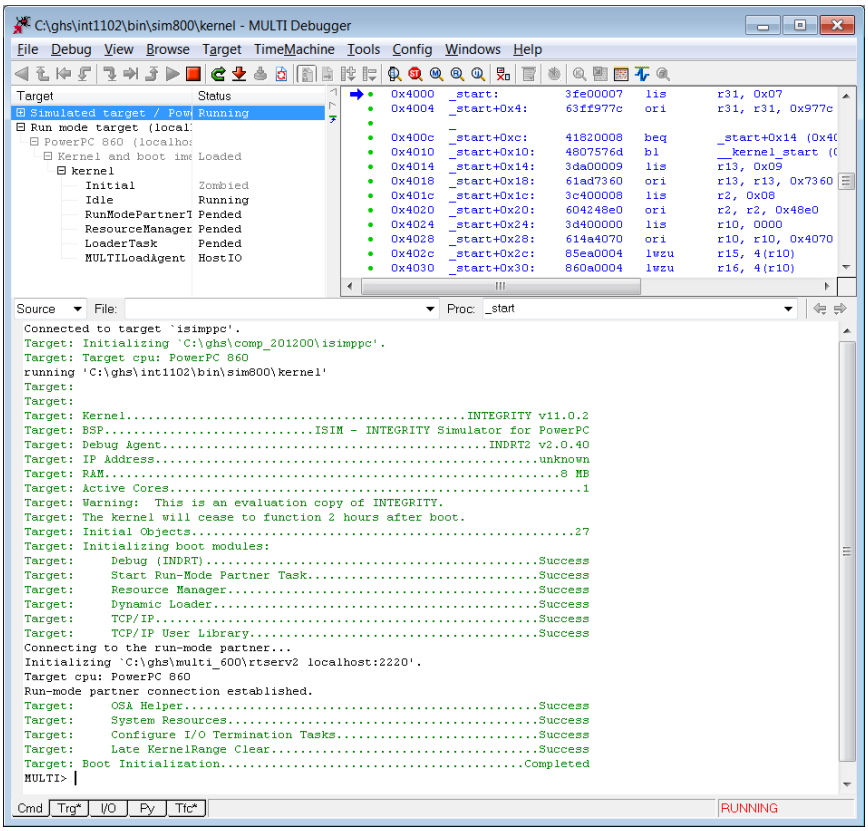
- Launches the MULTI Debugger on an INTEGRITY kernel.
- Connects to the appropriate INTEGRITY simulator.
- Instructs the Debugger to download the kernel to the simulator and start it running.
- Launches **rtserv2**, the debug server, and instructs it to connect to the running kernel.

These instructions assume that you have already created the MULTI Launcher workspace for the simulator of your choice as described in the “Creating MULTI Workspaces for Installed BSPs” section earlier in this chapter.

1. On the Launcher’s Action list, double-click **Run and Debug**.



Upon successful connection to the ISIM target, the run-mode debug Tasks will appear in the MULTI Debugger Window:



2. The first time you open the MULTI Debugger, the Target list is located at the top of the window, with the source pane below. INTEGRITY users should move the target list to the left side of the Debugger window (as shown in the picture above). To do this, click the **Move target list to left** button (), located to the right of the column headings. MULTI remembers the location of the Target list across sessions.

You have now completed the process. An INTEGRITY kernel is running on ISIM and you have connected the MULTI Debugger to the running kernel. If you are new to INTEGRITY, it is highly recommended that you complete the tutorials in the “Using INTEGRITY Tutorials” section of the *INTEGRITY Development Guide* before attempting to create your own application.

4.5 Target Board Setup

The instructions used in earlier sections of this chapter used an INTEGRITY Simulator (ISIM) to run the kernel on a simulated target. However, the process of loading your own application to an actual board is different than it is for ISIM.

To boot a kernel on a board, you may need to use a board-specific boot loader or ROM monitor, or you may be able to use a hardware debugger (for example, the Green Hills Probe) with an associated debug server (e.g., mpserv). For information about booting an INTEGRITY kernel or Monolith on your board, see the ***bspname*.notes** file for the BSP you are using.

To boot a kernel on an x86 or x64 PC target, you must have already set up the target to run INTEGRITY as its operating system. For information about setting up an x86 or x64 PC target to run INTEGRITY, see the *INTEGRITY GILA Boot Loader Guide*.

To develop code for an INTEGRITY system on an actual board, you will need:

- An INTEGRITY installation present on the host computer being used for development.
- A board set up and ready for use as described in this section.

Before attempting communication with the board, see the "Network Configuration" chapter of the *INTEGRITY Networking Guide*.

A typical board setup includes:

- A serial connection to the board from the user's host computer — the serial connection can be used to display RTOS diagnostics information and for Task debugging. Sometimes the serial connection is required in order to communicate with a board's factory installed firmware (if any).
- An Ethernet connection to the board — Ethernet can be used for Task debugging (most BSPs support this functionality) and for TCP/IP communication (provided with some INTEGRITY installations). If Ethernet is used, the system administrator must provide the board with a valid IP address for the local network. The *INTEGRITY Development Guide* describes more about the role of serial and Ethernet in the INTEGRITY environment.
- Other Hardware — depending on the type of freeze-mode debugging solution for a particular board, some hardware setup may be necessary. For example, on the Freescale CDS 8548, the suggested freeze-mode debug solution is a Green Hills Probe that is connected to the board's debug port. For more information, consult the "Freeze-Mode Debugging" chapter of the *INTEGRITY Development Guide*.
- BSP Specific Configuration — consult the ***bspname*.notes** file for the particular BSP you are using. This file contains important information about how to set up the board and load a kernel to it.
- x86 and x64 Specific Configuration — consult the *INTEGRITY GILA Boot Loader Guide* for additional information about setting up an x86 or x64 PC target.

Note: In the event that the board is either not yet available or in limited supply, ISIM enables programmers to develop and test embedded INTEGRITY-based applications without an actual board.

4.6 Booting an INTEGRITY Kernel

After a hardware connection has been established with the target board, an INTEGRITY kernel must be booted. The instructions used in earlier sections of this chapter used the MULTI Launcher **Run and Debug** action to boot a kernel on an INTEGRITY Simulator (ISIM). However, the process of booting a kernel on an actual board is different and depends on which board you are using, and whether the kernel was built for RAM or ROM.

- On some boards, you use factory-installed firmware and TFTP to boot the kernel over the network.
- On other boards, you boot the kernel through MULTI by using **mpserv** and a Green Hills Probe.
- On an x86 or x64 PC target, you use the GILA boot loader to boot the kernel.
- Other boards and/or custom designs may use entirely different booting methods.

Consult the ***bspname*.notes** file for the particular reference BSP you are using. This file often contains important information about how to set up the board and boot a kernel on it.

Each reference BSP provided in the INTEGRITY Installation includes a default kernel. You can either boot the default kernel into RAM on the target board, or you can build a custom kernel. For more information, consult the "Building INTEGRITY Applications" chapter of the *INTEGRITY Development Guide*.

The default kernel is located in the **bin*bspname*** directory. For example, if the INTEGRITY Installation is located in the **c:\ghs\int*version*** directory, then the default kernel for the ibm440 BSP would be **c:\ghs\int*version*\bin\ibm440\kernel**. For more information, see the "INTEGRITY Installation and Directory Structure" chapter of the *INTEGRITY Development Guide*.

If you use a loader that only works with files in a binary image (**.bin**) format, the Green Hills **elfloader** utility allows you to use ELF images with these loaders. Loaders that work only with files in the **.bin** format (such as PPCBug) can be wasteful because the **.bin** format includes a full section of zeros for any `bss` sections that precede the final section of the image. The `bss` sections of an ELF format file take up no space, and as a result, ELF produces smaller executables. For more information, see "elfloader" in the "Alternate Download Methods: elfloader and bootloader" chapter of the *INTEGRITY Libraries and Utilities User's Guide*.

Chapter 5

Appendix A: Third-Party Copyright Information

This chapter contains:

- DES Copyright Information
- The 4.4BSD Copyright
- The Free BSD Documentation License

5.1 DES Copyright Information

Copyright (C) 1995-1997 Eric Young (eay@cryptsoft.com) All rights reserved.

This package is an DES implementation written by Eric Young (eay@cryptsoft.com). The implementation was written so as to conform with MIT's libdes.

This library is free for commercial and non-commercial use as long as the following conditions are adhered to. The following conditions apply to all code found in this distribution.

Copyright remains Eric Young's, and as such any Copyright notices in the code are not to be removed. If this package is used in a product, Eric Young should be given attribution as the author of that the SSL library. This can be in the form of a textual message at program startup or in documentation (online or textual) provided with the package.

Redistribution and use in source and binary forms, with or without modification, are permitted provided that the following conditions are met:

1. Redistributions of source code must retain the copyright notice, this list of conditions and the following disclaimer.
2. Redistributions in binary form must reproduce the above copyright notice, this list of conditions and the following disclaimer in the documentation and/or other materials provided with the distribution.
3. All advertising materials mentioning features or use of this software must display the following acknowledgement: This product includes software developed by Eric Young (eay@cryptsoft.com)

THIS SOFTWARE IS PROVIDED BY ERIC YOUNG “AS IS” AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE AUTHOR OR CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

The license and distribution terms for any publically available version or derivative of this code cannot be changed. i.e. this code cannot simply be copied and put under another distrubution license [including the GNU Public License.]

The reason behind this being stated in this direct manner is past experience in code simply being copied and the attribution removed from it and then being distributed as part of other packages. This implementation was a non-trivial and unpaid effort.

5.2 The 4.4BSD Copyright

All of the documentation and software included in the 4.4BSD and 4.4BSD-Lite Releases is copyrighted by The Regents of the University of California.

Copyright 1979, 1980, 1983, 1986, 1988, 1989, 1991, 1992, 1993, 1994 The Regents of the University of California. All rights reserved.

Redistribution and use in source and binary forms, with or without modification, are permitted provided that the following conditions are met:

1. Redistributions of source code must retain the above copyright notice, this list of conditions and the following disclaimer.
2. Redistributions in binary form must reproduce the above copyright notice, this list of conditions and the following disclaimer in the documentation and/or other materials provided with the distribution.
3. Neither the name of the University nor the names of its contributors may be used to endorse or promote products derived from this software without specific prior written permission.

THIS SOFTWARE IS PROVIDED BY THE REGENTS AND CONTRIBUTORS “AS IS” AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE REGENTS OR CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY

OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

5.3 The Free BSD Documentation License

Copyright 1994-2004 The FreeBSD Project. All rights reserved.

Redistribution and use in source (SGML DocBook) and "compiled" forms (SGML, HTML, PDF, PostScript, RTF and so forth) with or without modification, are permitted provided that the following conditions are met:

1. Redistributions of source code (SGML DocBook) must retain the above copyright notice, this list of conditions and the following disclaimer as the first lines of this file unmodified.

2. Redistributions in compiled form (transformed to other DTDs, converted to PDF, PostScript, RTF and other formats) must reproduce the above copyright notice, this list of conditions and the following disclaimer in the documentation and/or other materials provided with the distribution.

THIS DOCUMENTATION IS PROVIDED BY THE FREEBSD DOCUMENTATION PROJECT "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE FREEBSD DOCUMENTATION PROJECT BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS DOCUMENTATION, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

Index