WIND RIVER

Wind River® Platforms

GETTING STARTED

3.2

Corporate Headquarters
Wind River Systems, Inc.
500 Wind River Way
Alameda, CA 94501-1153
U.S.A.

toll free (U.S.): (800) 545-WIND
telephone: (510) 748-4100
facsimile: (510) 749-2010

For additional contact information, please visit the Wind River URL:

http://www.windriver.com

For information on how to contact Customer Support, please visit the following URL:

http://www.windriver.com/support

Wind River Platforms Getting Started, 3.2

9 Nov 05
Part #: DOC-15658-ND-00

Contents

1 Overview

1.1 Introduction

This document provides information about building, configuring, migrating, and locating additional documentation for each of the following Wind River Platform products:

- Wind River Platform for Automotive Devices (Platform AD) 3.2
- Wind River Platform for Consumer Devices (Platform CD) 3.2
- Wind River Platform for Industrial Devices (Platform ID) 3.2
- Wind River Platform for Network Equipment (Platform NE) 3.2

Platform AD

Designed expressly for developers of car infotainment and telematic devices, Platform AD combines the VxWorks real-time operating system and powerful Workbench development tool suite with necessary graphics and connectivity

components to support car infotainment and telematic devices, development tools, and reference hardware, where available.

Platform CD

Designed expressly for developers of consumer electronic products, Platform CD combines the VxWorks real-time operating system and powerful Workbench development tool suite with necessary graphics, networking, security, and management components to support consumer devices, development tools, and reference hardware, where available.

Platform ID

Designed expressly for developers of products targeted for use in industrial environments, Platform ID combines the VxWorks real-time operating system and powerful Workbench development tool suite with necessary communications components to support industrial devices, development tools, and reference hardware, where available.

Platform NE

Designed expressly for developers of products targeted for use in enterprise class networking environments, Platform NE combines the VxWorks real-time operating system and powerful Workbench development tool suite with necessary communications, routing, and management components to support network infrastructure devices, development tools, and reference hardware, where available.

Terminology and Conventions

The following terms are used in this document:

host
: A computer on which the Wind River development tools run.

target
: A processor board that runs VxWorks (Wind River's real-time operating system) and applications developed with Wind River Workbench.

target server
: A service that runs on the host and manages communications between host tools (such as the VxWorks development shell, debugger, and browser) and the target system itself. One target server is required for each target.

Wind River registry
A Wind River service that keeps track of, and provides access to, target servers. One registry may serve a network, or registries may run on each host.

The following conventions are used in this document:

1. The root installation directory is identified as *installDir* in this document, but the environment variable **WIND_HOME** must be set to the root installation directory for Wind River Workbench to work.
2. A series of items to be selected from the GUI is denoted by **A > B > C**. The elements **A**, **B**, and **C** may be menu items, buttons, or tabs.
3. Pathnames that apply to both UNIX and Windows are shown with forward slashes (/).

1.2 Wind River Platforms Features

Each Wind River Platform is based on a set of development tools and software run-time components, at the core of which is the VxWorks real-time operating system. All Platforms extend VxWorks by adding essential networking, multimedia, management, and communications components. Additionally, all Platforms include an enhanced set of host tools, including compilers, static analysis, run-time analysis, and debugging tools.

Each Platform includes the following components:

- VxWorks 6.2, a high-performance, memory-protected real-time operating system.
- Wind River VxWorks Simulator 6.2, the VxWorks host-based target simulator.
- Wind River Workbench 2.4, a development suite which facilitates managing and building projects, establishing and managing host-target communication,

and running, debugging, and monitoring VxWorks applications. Wind River Workbench also includes the following:

  - Wind River System Viewer 4.7, a software logic analyzer that provides graphical representations of the dynamic interactions of elements of the system.
  - on-chip debugging capabilities, including support for the Wind River ICE SX and Wind River Probe emulators.

- Wind River Compiler 5.3.1, a compiler for C and C++ development.
- GNU compiler 3.3.2, a compiler for C and C++ development.
- Wind River CAN 1.5.1, a collection of platform independent CAN (Controller Area Network) device drivers that share a common API.
- Wind River CLI, Web, MIBway 4.4.2, a collection of run-time components that allows you to build integrated Web-based and command-line device management solutions that both share a common backplane to access device data.
- Wind River DCOM 2.3.2, Wind River's implementation of COM and distributed COM (DCOM) on VxWorks.
- Wind River Firewall 2.1, which provides IP filtering with stateful inspection, MAC (Media Access Control) filtering, logging at the IP (L3) and data-link (L2) layers, non-volatile (NV) storage interface, and sample Web screens.
- Wind River IPsec and IKE 3.2, which provides applications with standards-based, interoperable cryptographic security services at the IP layer.
- Wind River Learning Bridge 1.3.1, a transparent layer 2 Ethernet learning bridge.
- Wind River Media Library 4.1, which provides foundation graphics, video, and audio technology, and a framework in which you can develop standardized custom device drivers.
- Wind River Mobile IPv6 3.0, an implementation of the protocol for *Mobility Support in IPv6*, as specified in RFC3775 of the Internet Engineering Task Force. IPv6 mobility support (Mobile IPv6) allows an IPv6 node that is configured for mobility to move from one network link to another without losing its connection to another node or its general accessibility across the network.
- Wind River NAT 2.1, a full-featured implementation of Traditional NAT (Network Address Translation), which includes support for Basic NAT and

NAPT (Network Address and Port Translation) for use in routers, firewalls, DSL, and cable modems.

- Wind River Network Stack 3.0, a dual IPv4/IPv6 TCP/IP network stack.
- Wind River OPC 3.1.1, an implementation of the OPC Foundations's specifications for Data Access (version 2.05A), Alarms and Events (version 1.10), and Data eXchange (version 1.0).
- Wind River OSPF 3.1.1, Wind River's implementation of the Open Shortest Path First interior gateway routing protocol for IPv4 and IPv6 networks.
- Wind River PPP 2.2.1, source code for managing protocol components through the Point-to-Point Protocol (PPP). Bundled with Wind River PPP 2.2 are PPP over Ethernet (PPPoE), Remote Access Framework (RAF), Multilink PPP (ML-PP), and BACP/BAP support. These components allow for the establishment, authentication, and control of PPP connections.
- Wind River RADIUS Client 1.3, a full-featured implementation of the Remote Authentication Dial-In User Specification (RADIUS), which can be used in a wide variety of networking equipment, including remote access servers, broadband remote access servers, access concentrators, virtual private network (VPN) routers, aggregators, and firewalls. Wind River RADIUS Client supports a complete set of standards for authentication, accounting, and security.
- Wind River ScopeTools 5.4, run-time analysis tools including:
  - Wind River MemScope 3.6, an online memory analyzer
  - Wind River ProfileScope 4.8, a statistical profiler
  - Wind River StethoScope 7.8, a real-time data monitor

  And optionally including:

  - Wind River CoverageScope 2.3, a code coverage analysis tool
  - Wind River TraceScope 3.4, an execution-flow trace tool
- Wind River Security Libraries 1.1, which provides security components with standards-based, configurable security services. Wind River Security Libraries implements parts of OpenSSL, the open source SSL development toolkit.
- Wind River SNMP 10.0.2, a portable Simple Network Management Protocol (SNMP) engine. It provides a collection of ANSI C subroutines to support the operation of SNMPv1, SNMPv2c, SNMPv3, and Agent Extensibility (AgentX) protocols. Wind River SNMP also provides the Wind River MIB Compiler, which is a tool that allows you to implement Management Information Bases (MIBs) with Wind River SNMP.

- Wind River SSL 1.1, this component provides native Secure Sockets Layer (SSL) and Transport Layer Security (TLS) services. Wind River SSL implements a subset of OpenSSL, the open source SSL development toolkit.
- Wind River TIPC 1.2, an implementation of a major subset of the Transparent Inter Process Communication (TIPC) protocol. This product is based on publicly available source code that is also used for the open source implementation on Linux.
- Wind River USB 2.2.2, including both a Universal Serial Bus (USB) host stack and a peripheral stack. The USB host stack enables you to write your own USB host controller driver. The USB peripheral stack enables you to write your own USB target application or target controller driver.
- Wind River Web Services 1.3, which provides development and run-time facilities for creating client and server SOAP applications, as well as XML applications. SOAP (Simple Object Access Protocol) is a lightweight protocol based on XML and HTTP that is used to exchange information in a distributed environment, and to invoke methods on servers and other objects. Wind River SOAP can be used to exchange SOAP messages with any system supporting Web services, such as systems running Microsoft .NET (R) or Apache Axis.
- Wind River Wireless Ethernet Driver 2.2, Wind River's implementation of the IEEE 802.11 standard for wireless connectivity. This driver is an 802.11a/b/g wireless LAN driver based on the Atheros AR521x MAC (Media Access Controller) chip that supports Station (STA), Access Point (AP), and Independent Basic Service Set (IBSS) functionality for PCI interfaces on selected BSPs.
- Wind River Wireless Security 2.1, Wind River's implementation of the IEEE 802.1X-2001 standard. IEEE 802.1X is a security standard for port-based network access control. It secures wired or wireless networks against unauthorized access by requiring authentication with a central server before network access and data transmission are allowed. Wind River Wireless Security provides 802.1X authenticator and supplicant roles on VxWorks.

For complete version information for the components included in Wind River Platforms, check your installation log file located at *installDir***/setup.log**.

1.3 Wind River Platforms Documentation Guide

Documentation for Wind River Platforms falls into these general categories:

1.3.1 Wind River Platforms Documentation

Wind River Platforms documentation includes:

- *Wind River Platforms Getting Started* (this guide)
- *Wind River Platforms Release Notes*

The release notes include:

- the latest list of supported hosts and targets
- information on compatibility with older releases
- an outline of new features
- any caveats concerning the current release

1.3.2 General Documentation

Your product also includes the general component documentation for each feature. This section outlines the documentation available with this release.

NOTE: In some cases, the product release version (see *1.2 Wind River Platforms Features*, p.3) does not match the documented product version shown in this section. For example, this release includes Wind River VxWorks Simulator 6.2. However, the document provided with this release is the *Wind River VxWorks Simulator User's Guide, 6.1*. For additional documentation updates, see the release notes.

Operating System Documentation

Documentation for the VxWorks operating system includes the following guides:

- *VxWorks Application Programmer's Guide, 6.2*
- *VxWorks Kernel Programmer's Guide, 6.2*
- *VxWorks Command-Line Tools User's Guide, 6.2*
- *VxWorks Architecture Supplement, 6.2*
- *VxWorks BSP Developer's Guide, 6.0*
- *VxWorks Device Driver Developer's Guide, 6.0*
- *VxWorks Migration Guide, 6.2*
- *VxWorks Hardware Considerations Guide, 6.0*

In addition to the above guides, the following reference documentation is available:

- VxWorks API References, 6.2
  - *VxWorks Application API Reference*
  - *VxWorks Drivers API Reference*
  - *VxWorks Kernel API Reference Volume 1: Libraries*
  - *VxWorks Kernel API Reference Volume 2: Routines*

  (In addition to HTML, the API references are available in UNIX-style man pages.)
- VxWorks BSP Reference
- VxWorks Errno Code List

Host Tools Documentation

The following host tool documentation is available with this release:

- **Wind River Workbench**

  Documentation for the Wind River Workbench includes:

  - *Wind River Workbench User's Guide, 2.4*
  - *Wind River Workbench Migration Guide, 2.3*
  - *Wind River System Viewer User's Guide, 4.7*
  - *Wind River Workbench User Interface Reference, 2.4*

- **Wind River ScopeTools**

  Documentation for the Wind River ScopeTools includes:

  – *Wind River MemScope User's Guide, 3.6*
  – *Wind River ProfileScope User's Guide, 4.8*
  – *Wind River StethoScope for VxWorks User's Guide, UNIX version, 7.8*
  – *Wind River StethoScope for VxWorks User's Guide, Windows version, 7.8*
  – *Wind River CoverageScope for VxWorks User's Guide, 2.2* (optional)
  – *Wind River TraceScope for VxWorks User's Guide, 3.3* (optional)

NOTE: Each Wind River ScopeTools user guide (as listed above) includes a detailed getting started chapter with information that you will find necessary and useful when invoking the tool.

- **Wind River VxWorks Simulator**

  Documentation for the Wind River VxWorks Simulator includes:

  – *Wind River VxWorks Simulator User's Guide, 6.1*

- **Wind River Compiler**

  Documentation for the Wind River Compiler includes:

  – Wind River Compiler User's Guides, 5.3
  – Dinkum C++ Library Reference Manual

- **Wind River GNU Compiler**

  The GNU compiler documentation provided by Wind River is a convenient collection of the Free Software Foundation (FSF) manuals for the GNU C and C++ compiler and its supporting tools: the C preprocessor, assembler, static linker, binary utilities, and make utility. The documentation included in this release is as follows:

  – *Wind River GNU Binary Utilities, 3.3.2*
  – *Wind River GNU C Preprocessor, 3.3.2*
  – *Wind River GNU Make, 3.80*
  – *Wind River GNU Compiler: Using as, 3.3.2*
  – *Wind River GNU Compiler: Using the GNU Compiler Collection, 3.3.2*
  – *Wind River GNU Compiler: Using ld, 3.3.2*

NOTE: FSF develops software under Solaris/UNIX, and examples in their manuals reflect this. However, the GNU tools do operate reliably under Windows.

- **On-Chip Debugging**

  Documentation for on-chip debugging includes:

  – *Wind River Workbench On-Chip Debugging Guide, 2.4*
  – *Wind River Workbench On-Chip Debugging Command Reference, 2.4*
  – *Wind River Workbench On-Chip Debugging Configuration Options Reference, 2.4*
  – *Wind River ICE SX for Wind River Workbench Hardware Reference, 2.4*
  – *Wind River ICE SX Hardware Connection Quick Start, 1.0*
  – *Wind River Probe for Wind River Workbench Hardware Reference, 2.3*
  – *Wind River Probe Hardware Connection Quick Start*

Connectivity Documentation

Documentation for connectivity components includes:

- *Wind River CAN for VxWorks 6 Programmer's Guide, 1.5*
- *Wind River DCOM for VxWorks 6 User's Guide, 2.3*
- *Wind River Mobile IPv6 for VxWorks 6 Programmer's Guide, 3.0*
- *Wind River Network Stack for VxWorks 6 Programmer's Guide, 3.0*
- *Wind River OPC for VxWorks 6 User's Guide, 3.1*
- *Wind River PPP for VxWorks 6 Programmer's Guide, 2.2*
- *Wind River TIPC for VxWorks 6 Programmer's Guide, 1.2*
- *Wind River USB for VxWorks 6 Host Stack Programmer's Guide, 2.2*
- *Wind River USB for VxWorks 6 Peripheral Stack Programmer's Guide, 2.2*
- *Wind River Wireless Ethernet Driver for VxWorks 6 User's Guide, 2.2*

Security Documentation

Documentation for security components includes:

- *Wind River Firewall and NAT for VxWorks 6 User's Guide, 2.1*
- *Wind River IPsec and IKE for VxWorks 6 Programmer's Guide, 3.2*
- *Wind River RADIUS Client for VxWorks 6 User's Guide, 1.3*
- *Wind River Security Libraries for VxWorks 6 Programmer's Guide, 1.1*
- *Wind River SSL for VxWorks 6 Programmer's Guide, 1.1*
- *Wind River Wireless Security for VxWorks 6 User's Guide, 2.1*

Management Documentation

Documentation for management components includes:

- *Wind River CLI, Web, MIBway for VxWorks 6 Programmer's Guide, 4.4*
- *Wind River CLI, Web, MIBway for VxWorks 6 API Reference, 4.4*
- *Wind River SNMP for VxWorks 6 Programmer's Guide, 10.0*
- *Wind River SNMP for VxWorks 6 API Reference, 10.0*

User Interface Documentation

Documentation for user interface components includes:

- *Wind River Media Library for VxWorks 6 DDK Programmer's Guide, 4.1*
- *Wind River Media Library for VxWorks 6 SDK Programmer's Guide, 4.1*
- *Wind River Media Library for VxWorks 6 API Reference, 4.1*

Bridging and Routing Documentation

Documentation for bridging and routing components includes:

- *Wind River Learning Bridge for VxWorks 6 User's Guide, 1.3*
- *Wind River OSPFv2 for VxWorks 6 Programmer's Guide, 3.1*
- *Wind River OSPFv3 for VxWorks 6 Programmer's Guide, 3.1*

Web Services Documentation

Documentation for Web services components includes:

- *Wind River Web Services for VxWorks 6 Programmer's Guide, 1.3*

1.3.3 Context Sensitive Help

Help buttons in Wind River Workbench and Wind River System Viewer provide information on the component you are currently using.

From a Workbench view, pressing **F1** on Windows, **Ctrl-F1** on Linux, or the **Help** button on Solaris opens an infopop containing a brief description of the view, as well as links to related topics in the documentation. Pressing **F1** within an enabled area in System Viewer opens the appropriate help page.

System Viewer has additional ways to access context-sensitive help. Right-clicking a specific event within a tool takes you to the System Viewer User's Reference Event Dictionary information for that event or state stipple.

1.3.4 Eclipse Documentation

The *Eclipse Workbench User Guide* and *Eclipse Platform Plug-in Developer Guide* (produced by the Eclipse Foundation, not by Wind River) provide information about general Eclipse concepts and procedures, as well as guidelines for writing your own Eclipse plug-ins.

1.3.5 Online Documentation

This section describes the online documentation available with this release.

Online Manuals

This release includes a subset of the previously described manuals in HTML and PDF format. You can open the online manuals from the **Help > Help Contents** menu in Workbench, or by navigating to your Workbench installation directory, then navigating to *installDir*/**docs/index_component.html**. A full-text search facility is available within the Workbench help browser.

Online Help

See *1.3.3 Context Sensitive Help*, p.11.

Man Pages

UNIX-style man pages for API reference entries are available for Solaris hosts. Type **man** *functionName* to display the man page.

1.3.6 Third-Party Documentation

In addition to the Wind River Web Services documentation created by Wind River, open source documentation for the Expat XML Parser 1.95.8 and for the gSOAP 2.7.6b toolkit is also provided. The open source documentation for both is available on the Online Support Web site at:

http://www.windriver.com/support

1.3.7 Getting Documents in Hardcopy

Wind River Platforms ship with the following manuals in hardcopy:

- *Wind River Platforms Getting Started* (this manual)
- *Wind River Platforms Release Notes*

Other books associated with this product suite can be ordered from the Wind River Bookstore. Visit the Bookstore at the following URL (support login required):

http://www.windriver.com/windsurf/bookstore

Once you have logged into the support site, click the link for the online bookstore.

NOTE: Print-ready PDFs are available from the title page of the corresponding online document. Navigate to the proper document under **Help > Help Contents > Wind River Documentation**, then click the PDF icon to the right of the book title.

2

Creating Projects

2.1 Introduction

This chapter describes types of builds with which you might be familiar, suggests the Wind River Workbench project most suited to each type of build, and provides pointers to additional information.

NOTE: It is possible to set up projects and perform some development immediately following installation and prior to building your Platform. However, full project capabilities will not be available until after you have compiled your Platform component source. For more information, see *3. Compiling Platform Component Source*.

Starting from the Command Line: Setting Your Environment

To use the tools included with your platform from the command line, you need to configure some environment variables and other settings. The best way to do this is with the **wrenv** environment utility.

- On Solaris and Linux, open a shell and set your environment to access the Wind River host tools and Wind River Compiler.

  In your installation directory, run the following command:

  ```
  % ./wrenv.sh -p vxworks-6.2
  ```

  NOTE: If your shell configuration file (**.profile**, **.cshrc**, **.tcshrc**, and so forth) overwrites the environment each time a new shell is created, the above command may not work. If you find that you cannot invoke the Workbench tools after executing the above command, use the following command:

  ```
  % eval `installDir/wrenv.sh -p platform -o print_env -f shell`
  ```

  where *shell* is **sh** or **csh**, depending on the current shell program. For example:

  ```
  % eval `./wrenv.sh -p vxworks-6.2 -o print_env -f sh`
  ```

- On Windows, select **Start > Programs > Wind River > VxWorks 6.2 > VxWorks Development Shell**.

For more information about the **wrenv** utility, see *VxWorks Command-Line Tools User's Guide*.

Starting Wind River Workbench

To start Workbench on Windows, go to **Start > Programs > Wind River > Wind River Workbench 2.4 > Wind River Workbench 2.4** (if you chose the default program group during installation).

To start Workbench on Solaris or Linux, enter the following:

```
% cd installDir
% ./startWorkbench.sh
```

For introductory information and a tutorial, see *Wind River Workbench User's Guide*. This manual is available from the Workbench online help system:

1. With Wind River Workbench running, choose **Help > Help Contents**.
2. In the online help table of contents, choose **Wind River Documentation > Guides > Host Tools > Wind River Workbench User's Guide 2.4 VxWorks Version**.

Additional tutorials for this release are available at:

http://www.windriver.com/support

2.2 VxWorks Boot Loaders

VxWorks Boot Loader projects are used to create a VxWorks boot loader (a *boot ROM*) to boot and load a target with a VxWorks kernel.

Workbench Build

You must create and configure a VxWorks boot loader project in order to build a VxWorks boot loader (also known as a boot ROM image) from within Workbench.

- To create a VxWorks boot loader project, see *Wind River Workbench User's Guide: Boot Loader Project*.
- To configure and build your VxWorks boot loader, see *Wind River Workbench User's Guide: Build Properties and the Build Console.*

NOTE: In this release, you cannot edit boot ROM files from within Workbench.

Command-Line Build

To build a VxWorks boot loader on the command line, navigate to the correct directory for your BSP (for example, *installDir*/**vxworks-6.2/target/config/** *bspName*) and then use **make**.

For instructions on how to call **make** directly, see *VxWorks Command-Line Tools User's Guide: Building Kernel and Application Projects*.

2.3 VxWorks Kernel Images

VxWorks Image Projects (VIPs) are used to configure and build a VxWorks kernel image to boot your target. A VIP can be a complete application and can also contain projects of other types such as file system projects.

Workbench Build

To build a VxWorks kernel image within Workbench, you must create and configure a VxWorks Image Project.

- For instructions about creating VxWorks Image Projects, see *Wind River Workbench User's Guide: VxWorks Image Projects*.
- For configuration and build guidelines, see *Wind River Workbench User's Guide: Build Properties and the Build Console*.

VxWorks Image Project Options

During VIP creation, the project creation wizard offers you the choice to include one or more options through a check box selection list. The three options are:

- **Use IPv6 enabled kernel libraries**

  This option includes the binary IPv6-enabled kernel libraries.

NOTE: This choice *only* refers to the precompiled IPv6 component binaries. Checking the IPv6 enabled libraries option presented by the wizard will result in linker errors when you attempt to build your project. Do *not* check this option. If you intend to use IPv6, you *must* set the appropriate component and feature macros in your platform **config.mk** file (see *3.2 Customize the Default Configuration*, p.24).

- **Use System Viewer free kernel libraries**

  This option includes libraries that are built without Wind River System Viewer instrumentation.

NOTE: This option is not supported for Platform products.

- **Use source mode build**

  This option is associated with the VxWorks scalability profiles and is *not* required for a standard project build. (For more information on scalable OS

profiles, see *VxWorks Kernel Programmer's Guide: Kernel Images, Components, and Configuration*.)

NOTE: All of the kernel components included in your project must support one of the scalability profiles. If not, your build will revert from a source build to a binary build; Workbench will *not* issue a warning or error message for this change.

Building a VxWorks File System Project

A VxWorks File System Project allows you to link a compiled-in file system (such as ROMFS or dosFS) with your VxWorks kernel image.

For information about creating file system projects, see *Wind River Workbench User's Guide: ROMFS File System Projects*.

Command-Line Build

If you are familiar with building a VxWorks kernel image using the command-line build, you can continue to use this technique.

- For general build information, see *VxWorks Command-Line Tools User's Guide: Building Kernel and Application Projects*.

NOTE: The **wrconfig** utility is not available in this release.

- For details about build commands, see the **vxprj** and **cmpscriptLib** reference pages.

  These are available on Solaris by typing **man vxprj** and **man cmpscriptLib**, and on all host platforms from **Help > Help Contents > Wind River Documentation > References > Host Tools > Wind River Project Configuration Command Line API Reference**.

Adding a ROMFS Target File System to your Kernel Image

You can also use **vxprj** to build a ROMFS target file system into your VxWorks kernel image. For details, see *VxWorks Command-Line Tools User's Guide: Working with Projects and Components*.

2.4 VxWorks Kernel Applications

VxWorks Downloadable Kernel Module (DKM) projects are used to manage and build modules that will exist in the kernel space. DKM projects provide an infrastructure for loading, unloading, running, and debugging downloadable kernel modules and kernel applications.

Workbench Builds

To build a VxWorks kernel application from within Workbench, you must create and configure a Downloadable Kernel Module project.

- For details about creating Downloadable Kernel Module projects, see *Wind River Workbench User's Guide: VxWorks Downloadable Kernel Module Projects*.
- To configure and build your kernel module, see *Wind River Workbench User's Guide: Build Properties and the Build Console*.

Command-Line Builds

If you are familiar with building application **.o** files to be loaded into the kernel using the command-line build (as was done in Tornado 2.x), you can continue to use this technique, but you must first create a Downloadable Kernel Module project with your application sources and build your project in Workbench at least once (see above). Then, you can build your kernel module on the command line.

For information about building your kernel module on the command line, see *VxWorks Command-Line Tools User's Guide: Building Kernel and Application Projects*.

2.5 Real-time Process Applications

VxWorks Real Time Process (RTP) projects are used to manage and build modules that will exist outside of the kernel space. This project type allows you to build, run, and debug VxWorks RTP executables.

Workbench Builds

To build a real-time process (RTP) application from within Workbench, you must create and configure a Real Time Process project.

When building an RTP that will use a shared library, the shared library must be explicitly added to the RTP.

- For information about creating RTP projects, see *Wind River Workbench User's Guide: Real Time Process Projects*.
- To configure and build your RTP, whether standalone or to use a shared library, see *Wind River Workbench User's Guide: RTPs and Shared Libraries from Host to Target*.

Command-Line Builds

To build an RTP on the command line, you must first create an RTP project with your RTP sources and build your project in Workbench at least once (see above). Then, you can build your RTP on the command line.

When building an RTP that will use a shared library, the shared library must be explicitly added to the RTP.

For information about building RTPs on the command line, see *VxWorks Command-Line Tools User's Guide: Building Kernel and Application Projects*.

2.6 Shared Libraries

VxWorks Shared Library projects are used for libraries that are dynamically linked to VxWorks Real Time Process projects at run-time or to create subprojects that are statically linked to other project types at build time.

Workbench Builds

To build a shared library from within Workbench, you must create and configure a VxWorks Shared Library project.

- For information about creating Shared Library projects, see *Wind River Workbench User's Guide: VxWorks Shared Library Projects*.
- For information about configuring and building a shared library, see *Wind River Workbench User's Guide: RTPs and Shared Libraries from Host to Target*.

Command-Line Builds

To build a shared library on the command line, you must first create a VxWorks Shared Library project with your library sources and build your project in Workbench at least once (see above). Then, you can build your shared library on the command line.

For information about building shared libraries on the command line, see *VxWorks Command-Line Tools User's Guide: Building Kernel and Application Projects*.

2.7 User-Defined Projects

Wind River Workbench provides support for VxWorks User-Defined projects. With this type of project, you must provide and maintain your own build system, file system management, and so forth. However, the Workbench user interface provides support for tasks such as configuring your build utility and creating build targets using the Workbench GUI. (The VxWorks source code build is an example of a VxWorks user-defined project.)

For more information on user-defined projects, see *Wind River Workbench User's Guide: VxWorks User-Defined Projects*.

Run-time Libraries (Source Build)

The VxWorks source code build is an example of a VxWorks user-defined project. If you are entitled to the VxWorks source code product, you can build all OS and component run-time source files that are distributed with the product. There are some **.o** files, however, that cannot be built due to licensing restrictions.

For details, see *3.6 Compiling VxWorks OS Source*, p.41.

3 Compiling Platform Component Source

3.1 Introduction

After you have installed your Platform software components, you *must* compile them for your target architecture and toolchain before you can use them in a customized VxWorks image. This requires two distinct steps: specifying exactly *what* you want to compile, then executing the actual compilation.

Before you start compiling, you must specify which components to compile, and (optionally) change the compile-time options for any of the components in your Platform (see *3.2 Customize the Default Configuration*, p.24).

After you have customized your Platform, you can execute the actual compilation using either of the following methods:

- Wind River Workbench

  Using Workbench to compile the Platform source is the easiest way to compile platform source.

- Command line

  Compiling from the command line is useful when you want to automate the build.

3.2 Customize the Default Configuration

Your Platform provides a set of makefiles that simplify the process of compiling most components. This scheme uses two makefiles (one for kernel and one for user builds) that call a makefile for each component that you choose to compile. The benefit of this approach is that all of the platform-level build selections are aggregated into two configuration files, one for kernel and one for user builds. These configuration files are located in your installation.

For kernel builds, use:

installDir/**vxworks-6.2/config/***platform*/**config.mk**

For user (RTP) builds, use:

installDir/**vxworks-6.2/config/***platform*/**configRtp.mk**

Example 3-1 shows part of the file *installDir*/**vxworks-6.2/config/pne/config.mk**. There are two types of definitions that you can customize for your Platform:

- *Choosing which Components to Compile*, p.24
- *Choosing Compile-Time Features for Selected Components*, p.26

Choosing which Components to Compile

The first part of Example 3-1 identifies features to be included (or source components to create VxWorks component binaries for). If you do not plan to use a given component in your project, change the value from **true** to **false**.

Building a feature (component) does not require that you use it, nor does it automatically include the component in your VxWorks image project. Rather, it creates a component binary for your chosen architecture that you *can* include into a VxWorks image project. Instructions for configuring VxWorks to include components begin in *4. Configuring VxWorks to Include Components*.

Example 3-1 **Platform NE Component Definitions**

```
export COMPONENT_SNMP     = true
export COMPONENT_WM       = true
export COMPONENT_PPP      = true
ifeq ($(KERNEL_MAJOR_VERSION),5)
export COMPONENT_USB      = true
endif
export COMPONENT_COREIP   = true
export COMPONENT_WLAN     = true
export COMPONENT_DOT1X    = true
export COMPONENT_IPSEC    = false
export COMPONENT_BRIDGE   = true
export COMPONENT_NAT      = false
export COMPONENT_RADIUS   = true
export COMPONENT_OSPFv2   = true
export COMPONENT_XML      = true
export COMPONENT_TIPC     = true
export COMPONENT_FIREWALL = false
export COMPONENT_OSPFv3   = false
export COMPONENT_SSL      = true
export COMPONENT_SECURITY = true

##
# SNMP feature configuration macros
#
export FEATURE_SNMP_V3          = true
export FEATURE_SNMP_AGENTX      = true

##
# CORE IP feature configuration macros
#
export FEATURE_COREIP_ROUTER     = true
export FEATURE_COREIP_VIRTUAL    = false
export FEATURE_COREIP_IPV6       = false
export FEATURE_COREIP_IPFW_HOOKS = false
export FEATURE_COREIP_IPSEC      = false
export FEATURE_COREIP_MIP6_MN    = false

##
# OSPF feature configuration macros
#
export FEATURE_OSPFV2_FTP    = false
export FEATURE_OSPFV2_MIB    = true
export FEATURE_OSPFV3_FTP    = false
export FEATURE_OSPFV3_MIB    = true
```

NOTE: Not all Wind River Platforms embedded software components can be compiled for your target architecture using this method. Wind River CLI, Web, MIBway and Wind River Media Library cannot be built using this method. For more information on compiling those components, see the programmer's guide for that component.

Choosing Compile-Time Features for Selected Components

The second section in Example 3-1 defines feature-specific compile options for certain components. For example, you can compile the Wind River SNMP code as a standard SNMPv1/v2c agent, or you can compile the optional SNMPv3 features. To turn off a feature-specific compile option, change the value from **true** to **false**.

Kernel and User Build Options

Some components provide the option to compile the component (or part of it) as a kernel component or as a user mode component (shared library or RTP). The component sections in this chapter indicate (alongside the component macro name) whether you can compile the component as *kernel only*, or *kernel or user*.

Kernel mode libraries are built with the kernel mode build, and user mode libraries are built with the user mode build. These libraries are completely separate.

Specifying **comp-kernel** when you compile the component source allows the compiled library API to be called from within the kernel only. In this case, the library API is not callable from user space (RTPs or shared libraries) until the user side of the library is built.

Specifying **comp-usr** when you compile the component source allows the compiled library API to be called from user space only. In this instance, the library API is not callable from kernel space until the kernel side of the library is built.

NOTE: You should edit the **config.mk** makefile to add or remove components from a kernel build, and edit the **configRTP.mk** file to add or remove components from a user build.

For those components that run in the kernel and are callable from user mode, the majority of the component is built for kernel mode by passing **comp-kernel**, and there is a very thin user mode *shim* that must be built in user mode to cross the trap boundary. (The shim is built by passing **comp-usr** in the second build pass.)

NOTE: Both kernel mode and user mode builds are required for components that run in the kernel and are callable from user mode.

When you compile the component source, you must specify either **comp-kernel** or **comp-usr**. If you do *not* specify either one of these, **comp-kernel** is implicitly specified.

For more information about using components with a shared library or RTP, see the component documentation.

SNMP Kernel and User Builds

If you plan to use the SNMP AgentX subagent in user (RTP) applications, you must first build the SNMP components for kernel mode. That is, compile the component source using **comp-kernel** (which is implicit), then compile using **comp-usr**.

3.2.1 Connectivity Component and Feature Descriptions

Wind River CAN

Edit the CAN component macro in the file *installDir*/**vxworks-6.2/config/***platform*/**config.mk** or **configRTP.mk**. The following macro is available:

COMPONENT_CAN (*kernel or user*)
Compiles the Wind River CAN components.

Wind River DCOM

Edit the DCOM component macro in the file *installDir*/**vxworks-6.2/config/***platform*/**config.mk** or **configRTP.mk**. The following macro is available:

COMPONENT_DCOM (*kernel or user*)
Compiles the Wind River DCOM components.

Wind River Network Stack

Edit the Network Stack component and feature macros in the file *installDir*/**vxworks-6.2/config/***platform***/config.mk**. The following macros are available:

COMPONENT_COREIP (*kernel only*)
Compiles the Wind River Network Stack components.

FEATURE_COREIP_IPFW_HOOKS
Compiles the network stack with support for IP filtering. This is required for Wind River Firewall, Wind River NAT, and any other application that requires IP filtering.

FEATURE_COREIP_IPV6
Compiles the network stack as a dual-mode IPv4/IPv6 stack.

FEATURE_COREIP_ROUTER
Compiles the network stack as a router stack (with network routing components and features).

FEATURE_COREIP_VIRTUAL
Compiles the network stack as a router stack with *virtual stack* capability. This option requires that you set **export FEATURE_COREIP_ROUTER = true**.

FEATURE_COREIP_IPSEC
This feature is required if you plan to use IPsec and IKE.

In addition to the component and feature macros listed above, certain network stack features are enabled using compile flags that can be passed to make on the command line, or can be added as user build arguments in your Workbench project. The following build options are available in addition to the component and feature macros:

ADDED_CFLAGS+=-DFASTUDP
Enables the Wind River Fast UDP feature.

ADDED_CFLAGS+=-DSCTP
Builds the SCTP modules.

NOTE: When the stack is built with the **-DSCTP** flag set, IPv6 (the **-DINET6** flag) is not supported.

ADDED_CFLAGS+=-DSOCKET_VLAN
Builds the socket-based VLAN support into the socket backend code.

ADDED_CFLAGS+=-DSUBNET_VLAN
Builds in support for FreeBSD-style VLAN pseudo-interfaces, thus enabling subnet-based VLANs.

ADDED_CFLAGS+=-DVLAN_TAG
Builds the legacy VLAN tagging library. This feature also requires **-DSOCKET_VLAN** and **-DSUBNET_VLAN**.

ADDED_CFLAGS+=IGMPV3
Builds the IGMPv3 modules. (IGMPv2 modules are created using the default build.)

ADDED_CFLAGS+=MLDV2
Builds the MLDv2 modules. (MLDv1 modules are created using the default build.)

Wind River Mobile IPv6

Edit the component and feature macros in the file *installDir***/vxworks-6.2/config/***platform***/config.mk**. The following component and feature macros are required for Mobile IPv6 support:

COMPONENT_COREIP (*kernel only*)
Compiles the Wind River Network Stack components.

COMPONENT_IPSEC (*kernel or user*)
Compiles the Wind River IPsec and IKE components.

COMPONENT_SECURITY (*kernel only*)
Compiles the Wind River Security Libraries components.

FEATURE_COREIP_MIP6_MN
Compiles the network stack with support for Mobile IPv6 and compiles the MIP6 MN application.

FEATURE_COREIP_IPV6
Compiles the network stack as a dual-mode IPv4/IPv6 stack.

FEATURE_COREIP_IPFW_HOOKS
Compiles the network stack with support for IP filtering. This is required for Mobile IPv6.

FEATURE_COREIP_IPSEC
This feature is required for Mobile IPv6.

NOTE: If you compile Wind River Mobile IPv6 to use the router stack (that is, the **-DROUTER_STACK** compile flag or **export FEATURE_COREIP_ROUTER = true**), you *cannot* include static configuration and startup of Mobile IPv6 in your VxWorks build.

Wind River OPC

Edit the OPC component macro in the file *installDir*/**vxworks-6.2/config/***platform*/**config.mk** or **configRTP.mk**. The following macro is available:

COMPONENT_OPC (*kernel or user*)
Compiles the Wind River OPC components.

Wind River PPP

Edit the PPP component macro in the file *installDir*/**vxworks-6.2/config/***platform*/**config.mk**. The following macro is available:

COMPONENT_PPP (*kernel only*)
Compiles the Wind River PPP demonstration components. For more information, see *Wind River PPP for VxWorks 6 Programmer's Guide, 2.2*.

The following flag can be manually passed to make on the command line or added as a user build argument in your Workbench project:

ADDED_CFLAGS+=-DPPP_DEBUG
This debugging feature allows Wind River PPP to print debug messages and packet dumps to the target console.

Wind River TIPC

Edit the TIPC component macro in the file *installDir*/**vxworks-6.2/config/***platform*/**config.mk**. The following macro is available:

COMPONENT_TIPC (*kernel only*)
Compiles the Wind River TIPC (transparent inter-process communication) components.

Wind River USB

Edit the USB component macro in the file *installDir*/**vxworks-6.2/config/***platform*/**config.mk**. The following macro is available:

COMPONENT_USB (*kernel only*)
Compiles the Wind River USB components.

The following flags can be manually passed to make on the command line or added as user build arguments in your Workbench project:

ADDED_CFLAGS+=NET2280_DMA_SUPPORTED
Enables DMA transfer for the NET 2280 driver (currently the ISP 1582 driver does not support DMA transfer).

ADDED_CFLAGS+=BULK_RESET_NOT_SUPPORTED
Certain devices such as EagleTec do not handle mass storage reset properly. Defining this macro means that no mass storage reset should be issued to the device.

Wind River Wireless Ethernet Driver

Edit the Wireless Ethernet Driver component macro in the file *installDir*/**vxworks-6.2/config/***platform*/**config.mk**. The following macro is available:

COMPONENT_WLAN (*kernel only*)
Compiles the Wind River Wireless Ethernet Driver components.

3.2.2 Security Component and Feature Descriptions

Wind River Firewall

Edit the firewall component macro in the file *installDir*/**vxworks-6.2/config/***platform*/**config.mk**. The following macro is available:

COMPONENT_FIREWALL (*kernel only*)
Compiles the Wind River Firewall components.

NOTE: Because Wind River Firewall *requires* that the network stack be built both as a router stack and with IP filter hooks, **FEATURE_COREIP_IPFW_HOOKS** and **FEATURE_COREIP_ROUTER** are both dynamically set to **true** in the feature set for your Platform whenever you set **export COMPONENT_FIREWALL = true**.

Wind River IPsec and IKE

Edit the IPsec and IKE component macro in the file *installDir*/**vxworks-6.2/config/***platform*/**config.mk** or **configRTP.mk**. The following macro is available:

COMPONENT_IPSEC (*kernel or user*)
Compiles the Wind River IPsec and IKE components.

Edit the following IPsec and IKE feature macros in the file *installDir*/**vxworks-6.2/vx_components/ipsecike-3.2/config.mk**:

FEATURE_IPSEC_CERTIFICATES
Set to **true** to include support for IKE certificates using RSA signatures.

FEATURE_IPSEC_NO_CERT_CHAIN
Set to **true** to disable sending certificate chains. IKE peers will not send the entire certificate chain during Phase 1 negotiation, but will send only their public certificate. The peer will still accept and verify certificate chains sent by other peers.

FEATURE_IPSEC_COUNTERS_IPSEC
Set to **true** to include the IPsec counters subcomponent for gathering and printing operational statistics.

FEATURE_IPSEC_COUNTERS_IKE
Set to **true** to include the IKE counters subcomponent for gathering and printing operational statistics.

FEATURE_IPSEC_LOGGING
Set to **true** to include the IPsec event logging subcomponent.

FEATURE_IPSEC_SEND_INITIAL_CONTACT
Set to **true** to force an initial contact message to be sent during each IKE negotiation. Set to **false** in order to have an initial contact message sent with only the first IKE negotiation with a peer.

The initial contact message allows Wind River IPsec and IKE to inform a remote peer that all pre-existing security information should be discarded and new security information must be negotiated.

FEATURE_IPSEC_MEMORY_ROUTINES
Set to **true** to make use of a memory pool for Wind River IPsec and IKE to help reduce memory fragmentation. These routines also include features to enable debugging memory allocation issues.

FEATURE_IPSEC_PMTU
Set to **true** to enable PMTU message handling by IPSec.

FEATURE_IPSEC_QUEUING
Set to **true** to create a separate VxWorks task for sending IPsec packets. If set to **false**, the outgoing IPsec packets are processed by the application's task.

FEATURE_WRN_SUPPRESS_INBOUND_MONOTONIC_PADDING_CHECK
Set to **true** to allow non-monotonic padding. If this is set, then padding that is not a monotonic sequence (12345...) will not create an error condition. This may be required to interoperate with IPsec implementations that do not strictly follow the padding guidelines in RFC2406, section 2.4.

FEATURE_IKE_PASSIVE_RESPONDER
Set to **true** to force an IKE responder to defer protection suite renewal to the initiator. If an IKE responder hits its own soft lifetime/size before the initiator has renewed the PS, it will not initiate renewal on its own. This can cause problems when the initiator's soft lifetime/size is larger than the responder's hard lifetime/size. In this situation, the protection suite will be deleted when the responder hits its hard lifetime/size, and will have to be renewed by the original initiator. This can cause a loss of traffic while the IKE negotiations occur.

Set to **false** to allow the responder to become the initiator. When the responder hits its soft lifetime/size, it will initiate a new IKE negotiation. Note that the policies of this "new" initiator will now be followed. The smaller lifetime/size attribute will be sent to the peer, which should accept them. This will have the effect of the new initiator remaining as initiator. That is, the peer with the smaller lifetime/size values will become the de-facto initiator.

Setting **FEATURE_IKE_PASSIVE_RESPONDER** to **false** is to be considered experimental for this release of Wind River IPsec and IKE. With short lifetimes, a message extraction failure may occur during quick mode negotiations, causing renewal of protection suites to fail. IPsec and IKE still function, but the protection suite has to be re-initiated.

Wind River NAT

Edit the NAT component macro in the file *installDir*/**vxworks-6.2/config/** *platform*/**config.mk**. The following macro is available:

COMPONENT_NAT (*kernel only*)
Compiles the Wind River NAT components.

NOTE: Because Wind River NAT *requires* that the network stack be built both as a router stack and with IP filter hooks, **FEATURE_COREIP_IPFW_HOOKS** and **FEATURE_COREIP_ROUTER** are both dynamically set to **true** in the feature set for your Platform whenever you set **export COMPONENT_NAT = true**.

Wind River RADIUS Client

Edit the RADIUS client component macro in the file *installDir*/**vxworks-6.2/config/** *platform*/**config.mk**. The following macro is available:

COMPONENT_RADIUS (*kernel only*)
Compiles the Wind River RADIUS Client components.

The following flags can be manually passed to make on the command line or added as user build arguments in your Workbench project:

ADDED_CFLAGS+=-D__RADIUS_MIB__
Builds the MIB (SNMP Management Information Base) for the RADIUS client. This debugging feature allows Wind River USB to print debug messages and packet dumps to the target console.

ADDED_CFLAGS+=-D__EAP__
Enables *Extensible Authentication Protocol* (RFC# 2869: *RADIUS Extensions*).

ADDED_CFLAGS+=-D__RADIUS_ATTRIBUTE_VERIFICATION_DEBUG__
Enables the debug RADIUS attribute verification feature (which verifies attributes before sending them to the RADIUS server).

Wind River SSL

Edit the SSL component macro in the file *installDir*/**vxworks-6.2/config/** *platform*/**config.mk**. The following macro is available:

COMPONENT_SSL (*kernel only*)
Compiles the Wind River SSL components.

Wind River Security Libraries

Edit the Security Libraries component macro in the file *installDir***/vxworks-6.2/config/***platform***/config.mk**. The following macro is available:

COMPONENT_SECURITY (*kernel only*)
Compiles the Wind River Security Libraries components.

Wind River Wireless Security

Edit the Wireless Security component macro in the file *installDir***/vxworks-6.2/config/***platform***/config.mk**. The following macro is available:

COMPONENT_DOT1X (*kernel only*)
Compiles the Wind River Wireless Security components.

3.2.3 Management Component and Feature Descriptions

Wind River CLI, Web, MIBway

NOTE: Wind River CLI, Web, MIBway cannot be built using the methods described in this chapter.

For more information on Wind River CLI, Web, MIBway, see *Wind River CLI, Web, MIBway for VxWorks 6 Programmer's Guide*.

Wind River SNMP

Edit the SNMP component and feature macros in the file *installDir***/vxworks-6.2/config/***platform***/config.mk** or **configRTP.mk**. The following macros are available:

COMPONENT_SNMP (*kernel or user using AgentX*)
The component macro compiles the Wind River SNMP v1/v2c agent components. You can also set one (or both) feature macros to **true** to compile SNMPv3 or components.

FEATURE_SNMP_V3
This feature macro compiles the SNMPv3 agent components.

FEATURE_SNMP_AGENTX
This feature macro compiles the eXtensible agent (AgentX) components. AgentX is required for managing components or applications that run as real-time processes (RTPs).

3.2.4 User Interface Components and Features

NOTE: Wind River Media Library cannot be built using the methods described in this chapter.

For more information on Wind River Media Library, see the *Wind River Media Library for VxWorks 6 DDK Programmer's Guide* or the *Wind River Media Library for VxWorks 6 SDK Programmer's Guide*.

3.2.5 Bridging and Routing Components and Features

Wind River Learning Bridge

Edit the learning bridge component macro in the file *installDir*/**vxworks-6.2/config/***platform*/**config.mk**. The following macro is available:

COMPONENT_BRIDGE (*kernel only*)
Compiles the Wind River Learning Bridge components.

Wind River OSPF

Edit the OSPF component macros in the file *installDir*/**vxworks-6.2/config/***platform*/**config.mk**. The following macros are available:

OSPFv2 macros:

COMPONENT_OSPFv2 (*kernel only*)
Compiles the Wind River OSPFv2 components.

FEATURE_OSPF_FTP
Use this option to implement OSPF so that it gets its configuration data using FTP.

FEATURE_OSPF_MIB
Use this option to enable SNMP management of OSPFv2.

OSPFv3 macros:

COMPONENT_OSPFv3 (*kernel only*)
Compiles the Wind River OSPFv3 components.

FEATURE_OSPFV3_FTP
Use this option to implement OSPF v3 so that it gets its configuration data using FTP.

FEATURE_OSPFV3_MAPI
Use this option to enable SNMP management of OSPFv3.

3.2.6 Web Services Components and Features

Wind River Web Services

Edit the Web Services component macro in the file *installDir*/**vxworks-6.2/config/***platform*/**config.mk** or **configRTP.mk**. The following macro is available:

COMPONENT_XML (*kernel or user*)
Compiles the Wind River Web Services components.

3.3 Compiling the Platform Component Source (Workbench)

To compile the platform source using Wind River Workbench, create a user-defined project with the appropriate build properties.

Step 1: Create a User-Defined Project

1. Open Workbench and select **File > New > User-Defined Project**.
2. From the **Target Operating System and Version** list, select **Wind River VxWorks 6.2**, then click **Next**.
3. Provide a name for this project.
4. Select **Create Project at External Location**, then type (or click **Browse** and navigate to) *installDir*/**vxworks-6.2**, then click **Next**.

NOTE: At this point, you can click **Next** or **Finish**. Clicking **Next** allows you to enter additional required build properties. However, you can edit these properties at any time by right-clicking your project and selecting **Properties**.

5. Ensure that **User-Defined Build** is selected, then enter the following as your **Build Command:**

   ```
   make CPU=cpuType TOOL=toolChain comp-[kernel|user]
   ```

   For example:

   ```
   make CPU=PPC32 TOOL=diab comp-kernel
   ```

   The **comp-kernel** and **comp-usr** arguments indicate whether certain component libraries should be built as user libraries or kernel libraries. Only certain components can be built as user libraries. For information about which components can be built as user libraries, see *3.2 Customize the Default Configuration*, p.24.

NOTE: To identify your CPU and associated primary compiler, see Table 3-2.

6. Enter the following as your **Clean Project rule**:

   ```
   TARGET=rclean
   ```

7. Click **Finish**.

Step 2: Clean the Project

Right-click your project and select **Clean Project**.

Step 3: Build the Project

Right-click your project and select **Build Project**.

Once your build is complete, the Platform components you chose to build will be available for inclusion in a VxWorks image. For more information on configuring a VxWorks image, see *4. Configuring VxWorks to Include Components*.

3.4 Compiling the Platform Component Source (Command Line)

Step 1: Set Your Command Environment

Before you build the components, you must first set the appropriate environment variables. The easiest way to do this is with the **wrenv** utility. For example, change to your install directory and execute the following:

```
$ wrenv.sh -p vxworks-6.2
```

or

```
C:\> wrenv.exe -p vxworks-6.2
```

The **wrenv** utility properly sets the necessary environment variables and prepares your environment for VxWorks development. For more information on the **wrenv** utility, see *VxWorks Command-Line Tools User's Guide*.

On Windows you can use the Start menu shortcut:
Start > Programs > Wind River > VxWorks 6.2 > VxWorks Development Shell.

Step 2: Determine any Feature Specific Compiler Flags

Based on the compile-time features you have selected, determine if there are any **ADDED_CFLAGS** you need to manually pass to make (see *Choosing Compile-Time Features for Selected Components*, p.26). For example, if you are building a VxWorks image project with the RADIUS client and SNMP management, you need to append the make command with the following arguments:

```
ADDED_CFLAGS+=-D__RADIUS_MIB__
```

WARNING: All additional compile flags must be appended to the **make** command exactly (case-sensitive) as they are documented.

Step 3: Determine the Appropriate CPU and TOOL Values

The top-level makefile *requires* that you specify a valid **CPU** argument and a valid **TOOL** (toolchain) argument. The values you specify for **CPU** and **TOOL** are dependent on your target CPU architecture. For a list of supported values, see *3.6.4 Supported CPU and TOOL Values*, p.47.

Step 4: Execute make rclean (Removing Objects)

Once you are satisfied that your **CPU** and **TOOL** values are correct, and you have determined which build options you wish to use, change your working directory to *installDir*/**vxworks-6.2** and execute **make TARGET=rclean**, *before* executing **make**.

To perform the clean, execute *exactly* the same **make** command you do to build (see Step 5), but append it with **TARGET=rclean**. For example:

1. Change your working directory to *installDir*/**vxworks-6.2**. (The simplest way to do this is by using the environment variable **WIND_BASE**. On Solaris and Linux systems, this is **$WIND_BASE**; on Windows, **%WIND_BASE%**.)

```
%> cd $WIND_BASE
```

2. Then, execute the **make** command.

```
%> make CPU=PPC32 TOOL=diab comp-kernel TARGET=rclean [optional cflags]
```

This removes all of the files created by the top-level makefile without removing any of the libraries supplied with your BSP.

Step 5: Execute make

To build your Platform, be sure that your **CPU** and **TOOL** values are correct and that you have determined which build options you wish to use. Then, change your working directory to *installDir*/**vxworks-6.2** and execute **make** as follows:

```
%> make CPU=cpuVal TOOL=toolVal comp-[usr|kernel] [optional cflags]
```

For example:

```
%> make CPU=PPC32 TOOL=diab comp-kernel [optional cflags]
```

The **comp-kernel** and **comp-usr** arguments indicate whether certain component libraries should be built as user libraries or kernel libraries. If you do not specify either one, **comp-kernel** is specified implicitly.

NOTE: Only certain components can be built as user libraries. For information about which components can be built as user libraries, see *3.2 Customize the Default Configuration*, p.24.

Example 3-2 and Example 3-3 show sample commands for executing the top-level make. The examples use different sets of values for CPU, toolchain, and feature set, as well as different compile options.

NOTE: Some CPU and toolchain combinations that are supported for certain components may not be supported for all of the components. Such combinations *will* cause the top-level make to fail. For more information about supported CPU and toolchain combinations, see your release notes.

Example 3-2 **Sample make Command: PPC32**

This example is a typical **make** command for the Motorola MBX860 BSP:

```
%> make CPU=PPC32 TOOL=diab comp-kernel
```


Example 3-3 **Sample make Command: MIPS32**

This example is a **make** command for the MIPS **malta5kc_mips32sfle** BSP using the required added flags for the RADIUS client MIB, and building select components as user mode components.

```
%> make CPU=MIPS32 TOOL=sfdiable comp-usr ADDED_CFLAGS+=-D__RADIUS_MIB__
```

In Example 3-3, notice that the **diab** toolchain designator is prepended by **sf**, which means *software floating point*; and appended by **le**, which specifies *little-endian* byte ordering for this switchable CPU.

NOTE: Building all of the component libraries may take a considerable amount of time.

Once your build is complete, the Platform components you chose to build will be available for inclusion in a VxWorks image. For more information on configuring a VxWorks image, see *4. Configuring VxWorks to Include Components*.

3.5 Changing Components and Features after Compiling

If you want to change your component and compile-time feature selections *after* you have executed the top-level make, edit your component and feature selections, clean your project (**rclean**), and re-compile the Platform source.

3.6 Compiling VxWorks OS Source

The VxWorks OS and run-time libraries are provided as pre-compiled binaries. While you must compile the Platform run-time source before you can use it, this is not necessary for the VxWorks source. However, if your development situation

requires it, you can recompile the VxWorks OS source. The output of this build is a set of run-time libraries that you can use to create a VxWorks image project.

NOTE: The instructions in this section apply to both kernel and user builds unless otherwise noted.

Source Code Limitations

There are some limitations and restrictions you must be aware of when recompiling the VxWorks source:

- The source code must be built from the VxWorks 6.2 installation tree with the host tools and makefiles provided.
- There may be certain portions of the VxWorks object code for which source code has not been provided (see *File Information*, p.42).
- For the unmodified source code that is included on the CD, the resulting binaries built using the Wind River Compiler should match the binaries distributed on the CD.
- Modifications to the source code (when permitted) may not be covered by Wind River Customer Support.

You can build the VxWorks source using Workbench or from the command line.

File Information

The VxWorks source code does not include some source files that have been licensed by Wind River from third parties under restriction and therefore cannot be redistributed in source code format. Table 3-1 lists, for each architecture, the **.o** files that the source code will not build.

NOTE: The build procedure documented in this chapter copies the relevant **.o** files into your source build directory.

Table 3-1 **Files Not Built by the Source Code**

Platform	Files		
ARMARCH5, ARMARCH6, XSCALE	**_x_ads_basic.o**	**_x_ads_dmul_mull.o**	**_x_ads_fflt_clz.o**
	_x_ads_d2f.o	**_x_ads_drem.o**	**_x_ads_ffltll_clz.o**
	_x_ads_daddsub_clz.o	**_x_ads_drnd.o**	**_x_ads_fgeqf.o**
	_x_ads_dcheck.o	**_x_ads_dsqrt.o**	**_x_ads_fleqf.o**
	_x_ads_dcmpin.o	**_x_ads_dunder.o**	**_x_ads_fmul_mull.o**
	_x_ads_dcmp.o	**_x_ads_except.o**	**_x_ads_fnorm2_clz.o**
	_x_ads_ddiv.o	**_x_ads_f2d.o**	**_x_ads_fpconst.o**
	_x_ads_deqf.o	**_x_ads_faddsub_clz.o**	**_x_ads_fpinit.o**
	_x_ads_dfixll.o	**_x_ads_fcheck.o**	**_x_ads_frem.o**
	_x_ads_dfix.o	**_x_ads_fcmp.o**	**_x_ads_frnd.o**
	_x_ads_dfixull.o	**_x_ads_fdiv.o**	**_x_ads_fsqrt_clz.o**
	_x_ads_dfixu.o	**_x_ads_feqf.o**	**_x_ads_funder.o**
	_x_ads_dflt_clz.o	**_x_ads_ffixll.o**	**_x_ads_istatus.o**
	_x_ads_dfltll_clz.o	**_x_ads_ffix.o**	**_x_ads_retnan.o**
	_x_ads_dgeqf.o	**_x_ads_ffixull.o**	**_x_ads_status.o**
	_x_ads_dleqf.o	**_x_ads_ffixu.o**	**_x_ads_trapv.o**
PowerPC8xx, 85xx, 403, 405, 440	**arc32.o**	**fpcmp.o**	**gccMathLib.o**
	ceil32.o	**fpexp32.o**	**hyp32.o**
	dccMathLib.o	**fpfloo32.o**	**log32.o**
	dp32.o	**fphyp32.o**	**mod32.o**
	dpcmp.o	**fplog32.o**	**pow32.o**
	exp32.o	**fpmod32.o**	**sqrt32.o**
	floor32.o	**fppow32.o**	**trig32.o**
	fp32.o	**fpsqrt32.o**	
	fparc32.o	**fptrig32.o**	
	fpceil32.o	**func32.o**	
PowerPC60*x*	**atan.o**	**sqrt.o**	**trig.o**
	pow.o		
All Pentium BSPs	**aic7880Lib.o**		

3.6.1 Building VxWorks Source Code Modules (Workbench)

This section describes how to build the VxWorks OS source using Workbench.

Step 1: Back Up the VxWorks Archives

In a host shell, go to the library subdirectory of the target directory. Make a copy of the VxWorks 6.2 archive directory and files for the architecture you want to rebuild. For example, to back up your PowerPC files, type:

```
% cd installDir/vxworks-6.2/target/lib
% mkdir ppcBackup
% cp -r ppc ppcBackup/ppc
% cp libPPC*.a ppcBackup
% cp -r objPPC* ppcBackup
```

Step 2: Create a User-Defined Project

1. Open Workbench and select **File > New > User-Defined Project**.
2. From the **Target Operating System and Version** list, select **Wind River VxWorks 6.2**, then click **Next**.
3. Provide a name for this project.
4. Select **Create project at external location**.
5. Click Browse, then navigate to (or enter):

   installDir/**vxworks-6.2/target/src**

NOTE: For user builds, navigate to *installDir*/**vxworks-6.2/target/usr/src**.

6. Click **Next**. The **Build Support** page appears.
7. Ensure that **User-defined build** option is selected, then enter the following as your **Build Command:**

   make CPU=*cpuType* **TOOL=***toolChain*

   For example:

   ```
   make CPU=PPC32 TOOL=diab
   ```

NOTE: To identify your CPU and associated primary compiler, see Table 3-2.

8. Click **Finish**.

Step 3: Rebuild the VxWorks Source

In the **Project Navigator**, right-click your project and select **Build Project**.

Step 4: Rebuild with a Secondary Compiler (optional)

After a successful build with the primary compiler, if you have **INCLUDE_GCC_INTRINSICS** in your project (or intend to build your project with the GNU GCC compiler), you also need to build the project with the secondary compiler (as listed in Table 3-2).

NOTE: If you are building the user-side source, you do not need to perform a rebuild with the secondary compiler. Only the Wind River Compiler is supported for user side builds.

You can change the build command by changing the values of the user build arguments.

1. In the **User Build Arguments** field at the top of the **Project Navigator** pane, add the secondary compiler information. The information you type here is appended to the build command you created. For example, if you enter **TOOL=gnu** as a user build argument, the build command becomes:

   ```
   make -f Makefile CPU=PENTIUM2 TOOL=diab TOOL=gnu
   ```

2. In the **Project Navigator** toolbar, click the **Build** button to start building the project for your CPU and associated secondary compiler.

After a successful build with the secondary compiler, clear the **User Build Arguments** field to revert to the primary compiler value.

3.6.2 Building VxWorks Source Code Modules (Command Line)

This section describes how to build the VxWorks OS source from the command line.

Step 1: Set Your Command Environment

Before you build the components, you must first set the appropriate environment variables. The easiest way to do this is with the **wrenv** utility, for example:

```
$ wrenv.sh -p vxworks-6.2
```

or

```
C:\> wrenv.exe -p vxworks-6.2
```

The **wrenv** utility properly sets the necessary environment variables and prepares your environment for VxWorks development. For more information, see *VxWorks Command-Line Tools User's Guide*.

On Windows, you can use the Start menu shortcut as follows:
Start > Programs > Wind River > VxWorks 6.2 > VxWorks Development Shell.

NOTE: If you want to disable the GNU dependency in any BSP, modify the **config.h** file for the BSP to include the following line:

```
#undef INCLUDE_GNU_INTRINSICS
```

Step 2: Back Up the VxWorks Archives

In a host shell, go to the library subdirectory of the target directory. Make a copy of the VxWorks 6.2 archive directory and files for the architecture you want to rebuild. For example, to back up your PowerPC files, type:

```
% cd installDir/vxworks-6.2/target/lib
% mkdir ppcBackup
% cp -r ppc ppcBackup/ppc
% cp libPPC*.a ppcBackup
% cp -r objPPC* ppcBackup
```

Step 3: Build the Source Code

Go to *installDir*/**vxworks-6.2/target/src** and start the build by invoking the **make** command to build the sources for your **CPU** and **TOOL**.

The syntax for the **make** command is:

```
% cd installDir/vxworks-6.2/target/src
% make CPU=cpuName TOOL=primaryCompilerName
```

If you have **INCLUDE_GCC_INTRINSICS** in your project (or intend to build your project with the GNU GCC compiler) you need to run **make** twice for each CPU—once for the primary compiler and once for the secondary compiler. To identify your CPU and associated primary and secondary compilers, see Table 3-2.

NOTE: If you do not need **INCLUDE_GCC_INTRINSICS** (which allows you to load **gnu**-built objects onto a **diab**-built image) or if you are performing a user-side build, do not run the secondary build.

To build for your secondary compiler, the syntax is the same as for the primary compiler. For example, run both of the following:

```
% make CPU=PPC32 TOOL=diab
% make CPU=PPC32 TOOL=gnu
```

NOTE: You can build a debug version of the source by providing a **-g** flag against **ADDED_CFLAGS** and **ADDED_C++FLAGS** in *installDir***/vxworks-6.2/target/src/Makefile**. For example:

```
% make CPU=cpuName TOOL=compilerName ADDED_CFLAGS+=-g ADDED_C++FLAGS+=-g
```

3.6.3 Restoring Original Archives and Object Directories

In the VxWorks development shell, go to the library subdirectory of the target directory. Move the recompiled version of the VxWorks 6.2 archive directory by renaming it, and then restore the original archive directory from the copy you made in *Step 2:Back Up the VxWorks Archives*, p.46.

For example:

```
% cd installDir/vxworks-6.2/target/lib
% mv ppc ppcRef
% mv ppcBackup/ppc ppc
% mv ppcBackup/libPPC*.a .
% cp -rf ppcBackup/objPPC* .
```

In this example, all of the newly recompiled files are now located in the *installDir***/vxworks-6.2/target/lib/ppcRef** directory.

3.6.4 Supported CPU and TOOL Values

The source tree build system has been designed and tested to compile source code with a primary compiler only. Table 3-2 lists the primary compiler that Wind River uses to build the VxWorks 6.2 image for each architecture. The secondary compiler is provided for the application level only, but some run-time support is required. Therefore, the source tree build system builds only the directories necessary to support the secondary compiler.

NOTE: User (RTP) builds support the CPU options listed in Table 3-2. However, only the Wind River Compiler (**diab**) is supported.

Table 3-2 lists both the primary and secondary compilers by architecture.

Table 3-2 **CPU and TOOL Values by Architecture**

Architecture	CPU	Primary Compiler	Secondary Compiler
ARM Architecture Version 5 Processors	ARMARCH5	diab	gnu
ARM Architecture Version 5 Processors (little-endian)	ARMARCH5	diable	gnule
ARM Architecture Version 6 Processors	ARMARCH6	diab	gnu
Intel XScale	XSCALE	diab	gnu
PowerPC 60x, 7xx, 74xx, 82xx	PPC32	diab	gnu
PowerPC 40x, 440, 8xx, 85xx	PPC32	sfdiab	sfgnu
MIPS32 BE	MIPS32	sfdiab	sfgnu
MIPS32 LE	MIPS32	sfdiable	sfgnule
MIPS64 BE	MIPS64	diab	gnu
MIPS64 LE	MIPS64	diable	gnule
Pentium	PENTIUM	diab	gnu
Pentium	PENTIUM2	diab	gnu
Pentium	PENTIUM3	diab	gnu
Pentium	PENTIUM4	diab	gnu
SH 7750, 7770	SH7750 (SH32 for RTP builds)	diab	gnu
SH 7750, 7770 LE	SH7750 (SH32 for RTP builds)	diable	gnule
SIMNT	SIMNT (SIMPENTIUM for RTP builds)	diab	gnu
SIMSOLARIS	SIMSPARCSOLARIS	diab	gnu

Table 3-2 **CPU and TOOL Values by Architecture** (cont'd)

Architecture	CPU	Primary Compiler	Secondary Compiler
SIMLINUX	SIMLINUX (SIMPENTIUM for RTP builds)	diab	gnu

NOTE: For **SIMNT**, **SIMSPARCSOLARIS**, and **SIMLINUX**, the **libprocfs.a**, **libtffs.a**, **libusb.a**, and **libusb2.a** libraries are not supported; although they are built from source, they are not present on the distributed CD.

4 Configuring VxWorks to Include Components

4.1 Introduction

This chapter explains how to include each of the Platform run-time components in a VxWorks Image Project (VIP). You can configure your VxWorks image project to include components a variety of ways: you can use Workbench to include components into a VxWorks image project, you can use the **vxprj** command-line build facility, or you can manually edit your BSP configuration files to include or exclude components and then build VxWorks from the command line. When

choosing a method for configuring and building your system, consider the following criteria:

- Configuring and building a VxWorks image using Wind River Workbench or the **vxprj** command-line build facility provides an easy and accurate method of adding needed components and ensuring required dependencies are also included.
- Configuring and building a VxWorks image using the BSP configuration method involves editing text files that contain component listings and parameters for initialization, and calling the **make** utility to build a system image. This process affords you the ability to automate the generation and build of your VxWorks image project, but it *requires* that you manage component dependencies.

NOTE: This document describes the general method for customizing a BSP to include components. Detailed component level configuration macros and descriptions are documented in the respective component documentation (see *1.3 Wind River Platforms Documentation Guide*, p.7).

- In addition, the following platform run-time components can also be built and run as real-time processes (RTPs):
  - Wind River OPC
  - Wind River DCOM
  - Wind River Media Library
  - Wind River CLI, Web, MIBway
  - Wind River Web Services
  - Wind River SNMP (AgentX sub-agents)

  For more information about using these components as RTPs, see the respective component documentation.

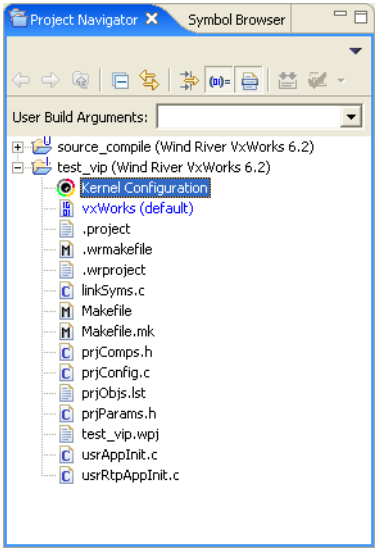
4.2 Locating Components

Each of the component inclusion procedures described is relative to the **Components** tab of the Kernel Editor. To view the Kernel Editor, you must have a VxWorks image project open in your workspace (see *2.3 VxWorks Kernel Images*, p.18).

To display the Kernel Editor:

1. In the Workbench **Project Navigator**, with your VxWorks image project expanded, double-click **Kernel Configuration** (Figure 4-1).
2. When the Kernel Editor displays, select the **Components** tab.

Figure 4-1 **Viewing the Kernel Editor**



There are a variety of ways in which you can select components to include in your VxWorks image project. The following sections include some examples.

4.2.1 Browsing the Kernel Editor

The **Components** tab of the Kernel Editor displays a listing of components grouped by functionality.

- **Bold Text**—identifies components that are already included in your project.
- Regular Text—identifies components *not* included in your project.
- *Italic Text* —identifies components that are not available for use in your project.

Components (and subcomponents) may not be available for use for a variety of reasons, including:

– The component source may not have been compiled to support the desired feature.

– The component may not be supported for use with your platform.

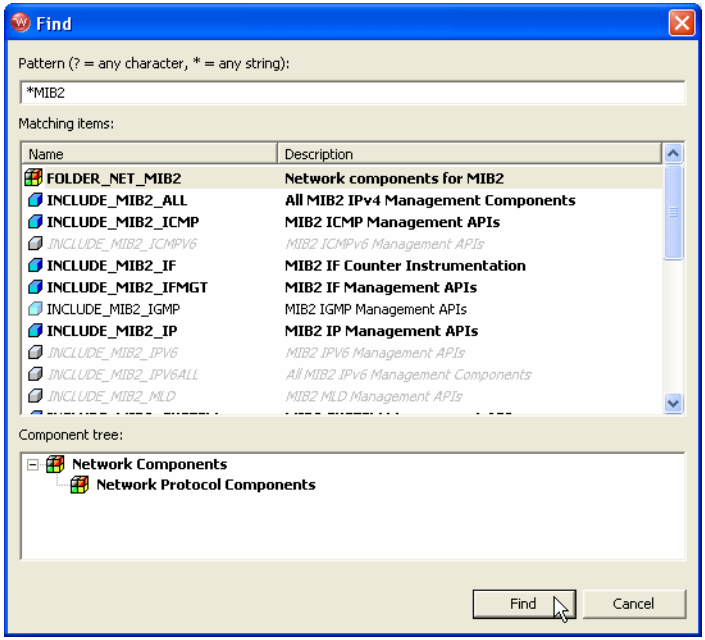
– The component may not be installed.

4.2.2 Using Find

The Find tool searches component **INCLUDE**_*ABC* macro names. If you know the component name (or part of it), you can use the Find tool to locate and select that component. The example in Figure 4-2, shows the matches found using the pattern ***MIB**.

When you see the component you want, select it and click **Find**. This returns you to the Kernel Editor with the component highlighted.

Figure 4-2 **Using Find**

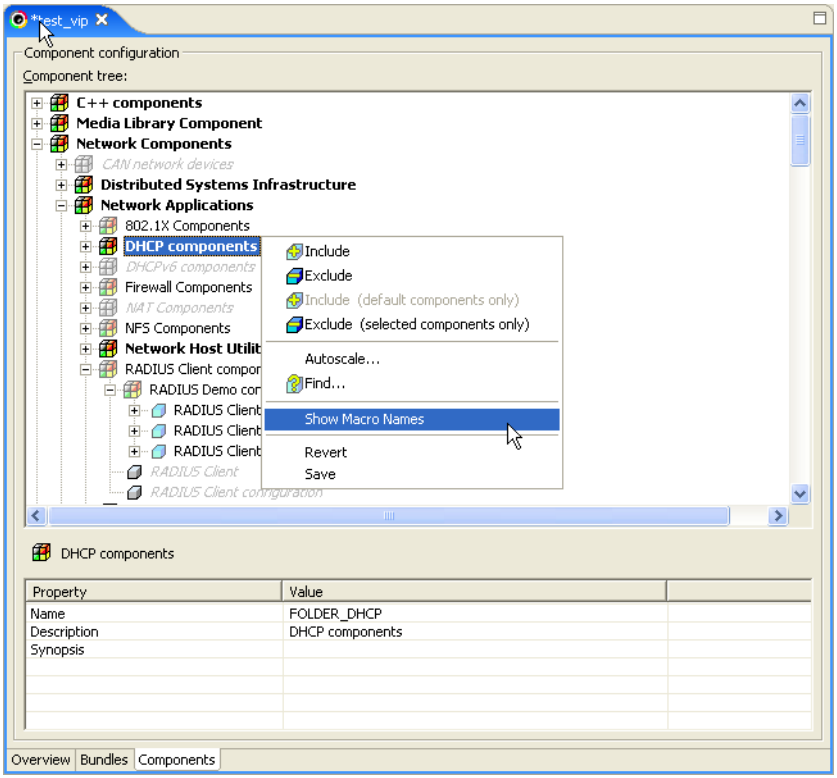


You can also browse through the component hierarchy in the Kernel Editor.

4.2.3 Show Macro Names

Workbench also provides a convenient way to show component **INCLUDE**_*ABC* macro names alongside their textual name. To show macro names, right-click any space in the Kernel Editor and select **Show Macro Names** (Figure 4-3).

Figure 4-3 **Show Macro Names**



4.3 Including Components: VxWorks Command-Line Builds

You can also include or exclude component features using a command-line build. There are two methods for performing VxWorks command-line builds:

- You can add or remove components or component bundles using the **vxprj** command-line build facility.
- You can edit the definitions in your BSP **config.h** file.

For more information on which command-line build methods are available or preferred for a given component, see the documentation for that component.

vxprj Build Method

Using **vxprj** for your command-line build helps ensure that any component dependencies are accounted for and provides a simple command-line interface. For more information on using the **vxprj** command-line build facility, see *VxWorks Command-Line Tools User's Guide*.

BSP config.h Build Method

You can control your VxWorks image configuration by including or excluding definitions in the target-specific configuration header file **config.h**. This file is located in the directory *installDir***/vxworks-6.2/target/config/***bspName*. This file contains definitions that apply only to a specific target, and can also redefine default definitions in **configAll.h** that are inappropriate for a particular target. For example, if a target cannot access a device controller at the default I/O address defined in **configAll.h** because of addressing limitations, the address can be redefined in **config.h**.

The **config.h** header file includes definitions for the following parameters:

- default boot parameter string for boot ROMs
- interrupt vectors for system clock and parity errors
- device controller I/O addresses, interrupt vectors, and interrupt levels
- shared memory network parameters
- miscellaneous memory addresses and constants

WARNING: Do *not* edit **configAll.h**.

Some platform components require manual editing beyond defining or undefining the component **INCLUDE**_*ABC* macro and supplying parameter macro values. For more information, see the individual component documentation.

4.4 Configuring OS Components

For details about configuring all of the VxWorks core OS run-time components, see *VxWorks Kernel Programmer's Guide*.

4.5 Configuring Connectivity Components

Wind River Platforms *connectivity* components include:

- Wind River CAN
- Wind River DCOM
- Wind River Network Stack
- Wind River Mobile IPv6
- Wind River OPC
- Wind River PPP
- Wind River TIPC
- Wind River USB
- Wind River Wireless Ethernet Driver

4.5.1 Wind River CAN

Step 1: Include CAN Components

1. Locate **CAN network devices**, and select **Include**.
2. Choose the appropriate CAN hardware component.

   Workbench will automatically add dependent components.
3. Include optional components such as **CAN Show Routines**.

Step 2: Configure the CAN Hardware Component

For more information about CAN hardware component parameters and descriptions, see *Wind River CAN for VxWorks 6 Programmer's Guide*.

Step 3: Verify CAN Hardware Settings

Some CAN hardware switches must be set to agree with the parameter setting made above. For more information about hardware settings, see *Wind River CAN for VxWorks 6 Programmer's Guide* and your CAN board documentation.

4.5.2 Wind River DCOM

Wind River DCOM provides considerable configuration flexibility. You can build DCOM components as kernel components or RTP projects. The project creation and build process varies depending on how you choose to configure the DCOM components. For complete project build and configuration instructions, see *Wind River DCOM for VxWorks 6 Programmer's Guide*.

4.5.3 Wind River Network Stack

The Wind River Network Stack is a full-featured dual mode (IPv4 and IPv4/IPv6) TCP/IP stack. The network stack provides a wide array of networking libraries and applications, some of which can be configured in different ways, and may require that you re-compile the source. For more information on the network stack components, as well as additional configuration options, see the *Wind River Network Stack for VxWorks 6 Programmer's Guide*.

For a description of compile-time network stack options, see *3.2.1 Connectivity Component and Feature Descriptions*, p.27.

Step 1: Include the Network Stack Components

1. Right-click **Network Components** and select **Include**.
2. Select the features you want to include, then click **Finish**.

4.5.4 Wind River Mobile IPv6

This section provides basic information on the components and parameters required to build Mobile IPv6 support into your VxWorks image. Detailed information for building and configuring Mobile IPv6 is provided in *Wind River Mobile IPv6 for VxWorks 6 Programmer's Guide, 3.0: Building VxWorks to Include Mobile IPv6*.

Step 1: Include the Wind River Mobile IPv6 Components

1. Include the MIPv6 component, **INCLUDE_MIPV6**. (Workbench includes the **INCLUDE_MIP6_MN** component automatically when you include **INCLUDE_MIPV6**.)

   The component has two parameters:

   MIPV6CTL_DEBUG_CFG
   If set to **TRUE**, debugging is on. By default, this parameter is **FALSE**.

   MIPV6CTL_USE_IPSEC_CFG
   If set to **TRUE**, this parameter enables use of IPsec. By default, this parameter is **FALSE**.

2. Include the mobile node component, **INCLUDE_MIP6_MN**. (Workbench includes the **INCLUDE_MIPV6** component automatically when you include **INCLUDE_MIP6_MN**.)

   This component provides parameters for statically configuring Wind River Mobile IPv6 and starting it when VxWorks starts. By default, the mobile node is set for dynamic configuration and startup.

NOTE: Currently, if you use static configuration, only one interface on the mobile node can be configured for mobility. Dynamic configuration allows you to configure multiple interfaces for mobility.

4.5.5 Wind River OPC

Wind River OPC provides considerable configuration flexibility. You can build OPC components as kernel components or RTP projects. The project creation and build process varies depending on how you choose to configure the OPC components. For complete project build and configuration instructions, see *Wind River OPC for VxWorks 6 Programmer's Guide*.

4.5.6 Wind River PPP

Wind River PPP provides a configurable framework for creating and managing various types of PPP connections. This section describes the basic steps required to implement a basic serial PPP connection framework. For full instructions for each connection type as well as instructions for creating and testing PPP connections, see *Wind River PPP for VxWorks 6 Programmer's Guide*.

Step 1: Increase the values of IP_MAX_UNITS

IP_MAX_UNITS indicates the maximum number of IP interfaces that the network stack will support. Each PPP connection that you create (serial or PPP over Ethernet) requires one IP interface. Locate the **IP_MAX_UNITS** parameter with the Find tool.

Step 2: Include the DNS resolver component in your image

1. Include **INCLUDE_DNS_RESOLVER.**
2. Within this component, set the following parameters:

   RESOLVER_DOMAIN
   For example: **"wrs.com"**

   RESOLVER_DOMAIN_SERVER
   For example: **"90.0.0.11"**

   If you do not know the correct values for these parameters, consult your system administrator.

NOTE: The quotation marks in the examples above are part of the value. You must include these quotation marks in whatever values you supply for the **RESOLVER_DOMAIN** and **RESOLVER_DOMAIN_SERVER** parameters.

Step 3: Include either the PPP client or the PPP server in your image

Add one of the following components:

INCLUDE_PPP_CLIENT
PPP client (DUN)—the PPP client

INCLUDE_PPP_RAS_CONC
Remote Access Server—the PPP server

If you are connecting two VxWorks targets, create two projects, one for each target. The kernel image for the client target should include the **INCLUDE_PPP_CLIENT**

component. The kernel image for the server target should include **INCLUDE_PPP_RAS_CONC**.

Workbench adds a number of other necessary components to your project at this stage.

The default PPP protocol framework is referenced by the variable **pppSysFramework** and its base configuration profile by **pppBasicLinkProfile**. This profile contains basic configuration information for a PPP stack but does not include any port-specific information.

Use the **pfwComponentListShow( )** routine to generate a listing of those modules that have been included in this framework. For example, from the target shell issue the following command:

```
-> pfwComponentListShow pppSysFramework
```

Step 4: Configure the serial connections (Null Modem Cable and External Modem)

This procedure references configuration parameters and components such as **INCLUDE_PPP_COM***n* and **PPP_COM***n***_PEER_TYPE**. The *n* refers to the number of the COM port, that is: COM1 or COM2.

When selecting a COM port, keep in mind that if you include the target shell in your kernel image, COM1 is already taken (for the target shell) and is not available for PPP. If you create PPP connections on COM1, you must exclude the target shell from your kernel image. To do this, locate the parameter **CONSOLE_TTY** using the Find tool and change it from the default of COM1 (**0**) to NONE (**-1**).

1. Include the **basic configuration for serial port** *n* component.

   INCLUDE_PPP_COM*n*
   basic configuration for serial port *n*

   When you include this component, the Kernel Editor will warn you of configuration errors. To fix this, you must set the value of the **PPP_COM***n***_PEER_TYPE** parameter as described in the next step.

2. Configure the **basic configuration for serial port** *n* component.

   Under the **Properties** tab, set the following parameters:

   PPP_COM*n***_PEER_TYPE**
   PPP_COM*n***_BAUDRATE**
   PPP_COM*n***_CLNT_AUTOSTART** (optional, for PPP clients only)

Set the parameter values as follows:

PPP_COM*n*_PEER_TYPE
Specify the type of connection by selecting one of the following values:

NOTE: When setting **PPP_COM*n*_PEER_TYPE**, the quotes around the value are part of the value. You must include them.

"WIN_CLIENT"
A Wind River PPP-server-to-Windows-Dial-up-Networking (DUN) connection over a null modem.

"WIN_SERVER"
A Wind River PPP-client-to-Windows-Remote-Access-Server (RAS) connection over a null modem.

"REGULAR"
A server-to-non-Windows or client-to-non-Windows connection (for instance Wind River PPP to Wind River PPP, Wind River PPP to Solaris, or Wind River PPP to Linux) over a null modem

"MODEM"
A Wind River PPP-client-to-server (for instance Windows RAS) connection over an external modem

This feature is supported in the demonstration client, which calls routines defined in: *installDir*/**vxworks-6.2/target/src/ppp/unsupported/pppModem.c**.

The demonstration client lets you dial out with **pppCom*n*Connect( )**, and hang up with **pppCom*n*Disconnect( )**. For details about these routines as well as the modem open, read, write, close, and **ioctl( )** interface, see *Wind River PPP for VxWorks 6 Programmer's Guide, 2.2*.

The demonstration server does not handle modem (dial-in) connections. This is not a limit of Wind River PPP, just of the server demonstration code.

PPP_COM*n*_BAUDRATE
If necessary, change the default value (**"19200"**) to match that of the peer. (The quotes are part of the value; you must include them.)

PPP_COM*n*_CLNT_AUTOSTART
For a client image only, this specifies whether the client automatically connects at boot time. By default, the demonstration client is configured to

automatically connect to the server at boot time. To prevent this, set this parameter to **FALSE**.

Note that if you prevent automatic boot-time connection, you must start the connection manually by calling **pppCom*n*Connect( )** after booting. For example:

```
-> pppCom2Connect
```

3. Configure IPv4 and IPv6 parameters.

   Include the following component:

   INCLUDE_COM*n*_IPV4_PARAMS
   ipv4 parameters for serial port *n*

   If you will be running IPv6 over the PPP connection, also include:

   INCLUDE_COM*n*_IPV6_PARAMS
   ipv6 parameters for serial port *n*

4.5.7 Wind River TIPC

Wind River TIPC provides a variety of build options and configuration parameters. For detailed information on building and configuring your VxWorks image to include TIPC support, see *Wind River TIPC for VxWorks 6 Programmer's Guide: Building VxWorks to Include Wind River TIPC*.

Step 1: Include the TIPC Components

1. Right-click
   Network Components > Network Protocol Components > TIPC components, and select **Include**.
2. Select the features you want to include and click **Finish**.

Step 2: Include Show Routines (Optional)

If desired, include
Network Components > Network Utility Components > Show Routine Components > TIPC show routines.

4.5.8 Wind River USB

Wind River USB provides both host controller components and peripheral (device) components. This section includes basic instructions for including USB components in a VxWorks image project. The USB components are VxWorks kernel components.

USB Peripheral Stack Components

1. For a basic USB setup, include the USB peripheral stack component:
   - **USB Peripheral Stack** (required) –**INCLUDE_USB_TARG**
2. Include a target controller driver:
   - one of the following (required) –
     - **NetChip NET2280 –INCLUDE_NET2280**
     - **PDIUSBD12 –INCLUDE_PDIUSBD12**
     - **PhilipsIsp1582 – INCLUDE_PHILIPS1582**
3. If needed, include an appropriate function driver:
   - **Keyboard Emulator** (optional) – **INCLUDE_KBD_EMULATOR**
   - **Printer Emulator** (optional) – **INCLUDE_PRN_EMULATOR**
   - **Mass Storage Emulator** (optional) – **INCLUDE_MS_EMULATOR** (this also requires the dosFs file system component)
4. To include device initialization at system start up, choose the appropriate device initialization component(s):
   - **Targ Keyboard Emulator Init** (optional) – **INCLUDE_KBD_EMULATOR_INIT**
   - **Targ Printer Emulator Init** (optional) –**INCLUDE_PRN_EMULATOR_INIT**
   - **Targ Mass Storage Emulator Init** optional) – **INCLUDE_MS_EMULATOR_INIT**
5. To include a code exerciser, include the USB testing tool:
   - **usbTool** (optional) –**INCLUDE_USBTOOL**

   The USB Tool performs all necessary USB driver initialization and therefore cannot be used if any of the initialization components (**INCLUDE_***XXX***_INIT** macros) are included.

USB Host Stack Components

This section describes the basic steps to include the USB host controller software components in your VxWorks image project. Your choices will depend on the particular USB host controller(s) you plan to use. For additional details and descriptions of configuration parameters for each controller type, see *Wind River USB for VxWorks 6 Host Stack Programmer's Guide*.

1. For a basic USB setup, include the USB host stack component:
   - **USB Host Stack** (required) –**INCLUDE_USB**

   Selecting the USB host stack component includes support for the USBD.
2. Include at least one of the host controller components:
   - **one or more of the following** (required) –
     - **EHCI** –**INCLUDE_EHCI**
     - **OHCI** –**INCLUDE_OHCI**
     - **UHCI** –**INCLUDE_UHCI**

   Selecting the **EHCI**, **OHCI**, or **UHCI** components includes modules for those types of host controllers, respectively. All host controller components require that the **USB Host Stack** component also be selected. More than one host controller can be present on the image at once.
3. To include initialization at system start for the USB host stack, include the host stack initialization component:
   - **USB Host Stack Init** (optional) –**INCLUDE_USB_INIT**
4. To include host controller initialization at system start up, include the appropriate host controller initialization component(s):
   - **EHCI Init** (optional) –**INCLUDE_EHCI_INIT**
   - **OHCI Init** (optional) –**INCLUDE_OHCI_INIT**
   - **UHCI Init** (optional) –**INCLUDE_UHCI_INIT**
5. If you intend to test a device, include one or more of the peripheral device components:
   - **Keyboard** (optional) –**INCLUDE_USB_KEYBOARD**
   - **Mouse** (optional) –**INCLUDE_USB_MOUSE**
   - **Printer** (optional) –**INCLUDE_USB_PRINTER**
   - **Speaker** (optional) –**INCLUDE_USB_SPEAKER**

- **Mass Storage - Bulk** (optional) –**INCLUDE_USB_MS_BULKONLY**
- **Mass Storage - CBI** (optional) –**INCLUDE_USB_MS_CBI**
- **END - Pegasus** (optional) –**INCLUDE_USB_PEGASUS_END**

Selecting any of the USB peripheral device components includes the corresponding driver module. These components require that the USB host stack be present on the VxWorks image.

6. To include device initialization at system start up, choose the appropriate device initialization component:
   - **Keyboard Init** (optional) –**INCLUDE_USB_KEYBOARD_INIT**
   - **Mouse Init** (optional) –**INCLUDE_USB_MOUSE_INIT**
   - **Printer Init** (optional) –**INCLUDE_USB_PRINTER_INIT**
   - **Speaker Init** (optional) –**INCLUDE_USB_SPEAKER_INIT**
   - **Bulk Mass Storage Init** (optional) –**INCLUDE_MS_BULKONLY_INIT**
   - **CBI Mass Storage Init** (optional) –**INCLUDE_MS_CBI_INIT**
   - **END - Pegasus IPv4 Initialization** (optional) – **INCLUDE_USB_PEGASUS_END_INIT**

   Selecting any of the device initialization components includes the corresponding driver module. These components require that the USB host stack be present on the VxWorks image.
7. To include a code exerciser, include the USB testing tool:
   - **usbTool** (optional) –**INCLUDE_USBTOOL**

   The USB testing tool performs all necessary USB driver initialization. Therefore, it cannot be used if any of the initialization components (**INCLUDE**_*XXX*_**INIT** macros) are included. For more information about initialization and initialization dependencies, see *Wind River USB for VxWorks 6 Host Stack Programmer's Guide*.

USB Components: Command-Line Configuration

The USB components can be configured into a VxWorks image using a command-line build. For complete command-line configuration instructions, see *Wind River USB for VxWorks 6 Peripheral Stack Programmer's Guide* and *Wind River USB for VxWorks 6 Host Stack Programmer's Guide.*

4.5.9 Wind River Wireless Ethernet Driver

This section includes basic instructions for including Wind River Wireless Ethernet Driver components in a VxWorks image project. These are VxWorks kernel components. For more detailed instructions and advanced configuration options, see the *Wind River Wireless Ethernet Driver for VxWorks 6 User's Guide*.

NOTE: Not all BSPs include support for the Wireless Ethernet Driver. For more information on adding support to your BSP, see *5.4.8 Wind River Wireless Ethernet Driver*, p.89.

1. Right-click **Network Components > Network Device Components > 802.11a/b/g Components** and select **Include**.
2. Select **common components** and include it in your project.
3. Under **802.11a/b/g Components**, expand **wireless hardware support** to include the Atheros chipset support.
4. Under **802.11a/b/g Components**, expand **wireless mode support** to include the wireless modes you wish to support in your project build. At least one mode must be selected. The driver can operate in three different modes to address all aspects of the IEEE 802.11 specification. These modes are:
   - **access point support**
   - **ad-hoc (IBSS) support**
   - **station (ESS) support**

NOTE: If you include the **access point support** mode, you must also include a learning bridge component. For basic instructions to include Wind River Learning Bridge, see *4. Configuring VxWorks to Include Components*.

5. If WPA/802.11i support is required, then the Security Libraries component is required. If WPA/802.11i enterprise mode support is required, then the Wireless Security component is required.
6. Right-click **802.11a/b/g Components**, select **Properties**, and then the **Params** tab. These parameters are default wireless network parameters that will allow basic wireless network connectivity at system boot time without requiring additional run-time configuration. These parameters are primarily designed for development and testing purposes and include security or power

management features as well. Configure the following wireless network parameters as desired:

```
DOT11_DEFAULT_SSID     /* SSID (Network Name)                          */
DOT11_DEFAULT_MODE     /* ESS, IBSS or AP mode                         */
DOT11_DEFAULT_RADIO    /* 802.11a, 802.11b, 802.11g or other modes     */
DOT11_DEFAULT_CHANNEL  /* required for IBSS and AP modes               */
DOT11_DEFAULT_COUNTRY  /* The country code to use that determines      */
                       /* channels                                     */

DOT11_AUTH_PSK_KEY     /* The default Pre-shared Key used if 802.11i   */
                       /* or WPA uses PSK as the Authentication method*/
DOT11_AUTH_PSK_PASSPHRASE /* The passphrase used to generate the PSK   */
                          /* if INCLUDE_DOT11_AUTH_PSK_PASSPHRASE is   */
                          /* selected                                  */
```

For more information on how to set these values, refer to the command-line build definitions in *installDir*/**vxworks-6.2/target/src/drv/wlan/bsp/sysDot11End.h.**

Wireless Ethernet Driver Components: Command-Line Configuration

The Wireless Ethernet Driver components can be configured into a VxWorks image using a command-line build. For complete command-line configuration instructions, see the *Wind River Wireless Ethernet Driver for VxWorks 6 User's Guide.*

4.6 Configuring Security Components

Wind River Platforms *security* components include:

- Wind River IPsec and IKE
- Wind River Firewall
- Wind River NAT
- Wind River RADIUS Client
- Wind River SSL
- Wind River Security Libraries
- Wind River Wireless Security

4.6.1 Wind River IPsec and IKE

This section includes basic instructions for including IP security and Internet Key Exchange (IPsec and IKE) components in a VxWorks image project. The IPsec and IKE components are VxWorks kernel components, but you can link RTP projects against IPsec and IKE. For more information about linking with RTP projects, and full descriptions of configuration parameters, see *Wind River IPsec and IKE for VxWorks 6 Programmer's Guide*.

To include IPsec and IKE components:

1. Right-click **Network Components > Network Security Components** and select **Include**.
2. At a minimum, you must include:

- **IP Security (IPsec)**
- **Security Association Database (SADB)**
- **Security Policy Database (SPD)**
- **Manual Key Manager (MKM)**

You can optionally include:

- **Internet Key Exchange (IKE)**

Once you have located the component, right-click the component name and select **Include** to include it in your customized VxWorks image. Workbench will automatically calculate dependencies and prompt you to accept the inclusion of dependent components.

IKE can be excluded if your project only requires MKM. All other selected components are required.

After you have included the IPsec and IKE components, you can modify IPsec and IKE configuration parameters in the **Properties** pane.

Additional Component Requirements

Enabling certain configuration parameters for Internet Key Exchange requires that you include extra dependencies manually as follows:

- If you enable the **IKE_CERT_ENABLE** parameter, you must also include the X.509 Certificate Support (**INCLUDE_CERT_SUPPORT**) component and its dependencies.
- If you enable the **IKE_CERT_RAMDISK_ENABLE** parameter, you must also include the RAM disk driver (**INCLUDE_RAMDRV**) component and its dependencies.

IPsec and IKE Components: Command-Line Configuration

The IPsec and IKE components can be configured into a VxWorks image using a command-line build. For complete command-line configuration instructions, see *Wind River IPsec and IKE for VxWorks 6 Programmer's Guide*.

4.6.2 Wind River Firewall

This section includes basic instructions for including firewall components in a VxWorks image project. The firewall components require that your BSP support at least two network interfaces. Some BSPs may need to be modified to support additional interfaces. For more information about adding interfaces, see *Wind River Firewall and NAT for VxWorks 6 Programmer's Guide*.

To include firewall components:

1. Right-click **Network Components > Network Applications > Firewall Components** and select **Include**.
2. Include the appropriate filtering component for your project needs (IP filter or MAC filter), and include any optional firewall components:
   - **Firewall IP Filter**
   - **Firewall IP Filter Utilities**
   - **Firewall Logging**
   - **Firewall MAC Filter**
   - **Firewall Non-Volatile Storage Interface**
   - **Firewall Sample Web Screen**

NOTE: The **Firewall Sample Web Screen** component requires the **Wind River Web** component.

Wind River Firewall Components: Command-Line Configuration

The firewall components can be configured into a VxWorks image using a command-line build. For complete command-line configuration instructions, see *Wind River Firewall and NAT for VxWorks 6 Programmer's Guide.*

4.6.3 Wind River NAT

This section includes basic instructions for including NAT components in a VxWorks image project. The NAT components require that your BSP support at least two network interfaces. Some BSPs may need to be modified to support additional interfaces. For more information about adding interfaces, see *Wind River Firewall and NAT for VxWorks 6 Programmer's Guide*.

To include NAT components:

1. Right-click **Network Components > Network Applications > NAT Components** and select **Include**.
2. Include the required NAT components:
   - **NAT Agent**
   - **NAT Timer Facility**
3. Include optional components.

   Include the NAT management information base (MIB), show routines, and application layer gateway (ALG) components as required by your project. For details about these optional components, see *Wind River Firewall and NAT for VxWorks 6 Programmer's Guide*.

   - **FTP ALG**
   - **H.323 ALG**
   - **IPsec Passthrough ALG**
   - **MIB2 NAT**
   - **NAT Show Routines**
   - **PPTP Passthrough ALG**
   - **SIP ALG**

Wind River NAT Components: Command-Line Configuration

The NAT components can be configured into a VxWorks image using a command-line build. For complete command-line configuration instructions, see *Wind River Firewall and NAT for VxWorks 6 Programmer's Guide*.

4.6.4 Wind River RADIUS Client

This section includes basic instructions for including the RADIUS client in a VxWorks image project. The RADIUS client can be configured in many different ways, some of which require that you re-compile the source. For more information about additional configuration options, including RADIUS client with PPP, see *Wind River RADIUS Client for VxWorks 6 Programmer's Guide*.

1. Right-click
   Network Components > Network Applications > RADIUS Client components and select **Include**.
2. Include the required RADIUS client components:
   - **INCLUDE_RADIUSC** (**RADIUS Client**)
   - **INCLUDE_RADIUSC_CONFIG** (**RADIUS Client configuration**)
3. If needed, include the **RADIUS Client Demo configuration** components. The default configuration of these components can be changed to suit your needs.
   - **INCLUDE_RADIUSC_DEMO_CFG** (**RADIUS Client Demo configuration**)
   - **INCLUDE_RADIUSC_PRIMARY_SERVER_CFG** (**RADIUS Client Primary Server configuration**)
   - **INCLUDE_RADIUSC_SECONDARY_SERVER_CFG** (**RADIUS Client Secondary Server configuration**)

Wind River RADIUS Client Components: Command-Line Configuration

The RADIUS client components can be configured into a VxWorks image using a command-line build. For complete command-line configuration instructions, see the *Wind River RADIUS Client for VxWorks 6 Programmer's Guide*.

4.6.5 Wind River SSL

This section includes basic instructions for including Wind River SSL in a VxWorks image project. Wind River SSL provides native SSL and Transport Layer Security (TLS) services to Wind River Platforms components.

To include Wind River SSL in your VxWorks image project:

1. Right-click
   Network Components > Network Security Components > SSL/TLS and select **Include**.

2. Include the required components. At a minimum you must include:
   - **SSL/TLS**

   You can optionally include:
   - **SSL/TLS Applications**

Wind River SSL Components: Command-Line Configuration

The SSL components can be configured into a VxWorks image using a command-line build. For complete command-line configuration instructions, see *Wind River SSL for VxWorks 6 Programmer's Guide.*

4.6.6 Wind River Security Libraries

This section includes basic instructions for including Wind River Security Libraries in a VxWorks image project.

To include Security Libraries in your VxWorks image project:

1. Right-click **Security Libraries** and select **Include**.
2. Select the required components.
   - For the Cryptography library, include **Common Cryptographic Interface (CCI)**.
   - For the Digital Certificates library, include **X.509 Certificate Support**.

   Once you have located the component, right-click the component name and select **Include** *componentName* to include it in your customized VxWorks image. Workbench automatically calculates dependencies and prompts you to accept the inclusion of dependent components.

Wind River Security Libraries Components: Command-Line Configuration

The Security Libraries components can be configured into a VxWorks image using a command-line build. For complete command-line configuration instructions, see the *Wind River Security Libraries for VxWorks 6 Programmer's Guide.*

4.6.7 Wind River Wireless Security

Wind River Wireless Security provides functionality for both the 802.1X *authenticator* and *supplicant* roles. Configuration of each role is described in the following sections.

Including Authenticator Components

The Wireless Security authenticator components require that you also configure your VxWorks image project to include:

- Wind River RADIUS Client
- Wind River Learning Bridge

1. Right-click
   Network Components > Network Applications > 802.1X Components > 802 .1X Authenticator > Common configuration and select **Include**.
2. Set Authenticator Parameters
   - **DOT1X_AUTH_CONTROLLED_INTERFACE_NAME**—name of the 802.1X authenticated interface being controlled by the authenticator.
   - **DOT1X_AUTH_CONTROLLED_INTERFACE_NUM**—number of the 802.1X authenticated interface being controlled by the authenticator.
   - **DOT1X_AUTH_LEGACY_WEP_TYPE**—WEP key size to be used when in 802.1X-only mode. In WPA/802.11i mode, this setting has no effect. Set this parameter to **DOT1X_AUTH_40BIT_KEY** (1) or **DOT1X_AUTH_104BIT_KEY** (2).
   - **DOT1X_AUTH_OPERATING_MODE**—mode of operation, wireless (1) or wired (2). Selecting wireless mode will bind the Wind River Wireless Ethernet Driver to the 802.1X authenticated interface, and requires the Wind River Wireless Ethernet Driver to be installed, compiled, and configured.
   - **DOT1X_AUTH_UNCONTROLLED_INTERFACE_NAME**—name of the network interface being used to communicate with the 802.1X authentication server.
   - **DOT1X_AUTH_UNCONTROLLED_INTERFACE _NUM**—number of the network interface being used to communicate with the 802.1X authentication server.

Including Supplicant Components

The Wireless Security supplicant components require that you also configure your VxWorks image project to include either the Wind River Wireless Ethernet Driver or a wired Ethernet driver.

1. Right-click
   Network Components > Network Applications > 802.1X Components > 802 .1X Supplicant > Common configuration and select **Include.**

   The following components may be dependencies of the supplicant (depending on which EAP support you choose) and are included automatically:

   - **Network Components > Network Security Components > SSL/TLS**
   - **Security Library Components > X.509 Certificate Support**
   - **Security Library Components > Common Cryptographic Interface (CCI)** nd necessary sub-components)
   - **development tool components > show routines > memory show routine**

   ⚠ **CAUTION:** Including the Wireless Ethernet Driver security components before including the 802.1X supplicant can cause a known problem. This problem appears as an exception upon the initialization of the supplicant component. If you encounter this problem, you can work around it by manually including all items under
   Security Library Components > Common Cryptographic Interfaces > Nativ e Algorithm Support.

2. Set the supplicant common configuration parameters.

   - **DOT1X_SUPP_INTERFACE_NAME**—name of the interface to attach to EAPOL.
   - **DOT1X_SUPP_INTERFACE_NUM**—index number of the interface to attach to EAPOL.
   - **DOT1X_SUPP_PACKET_TRACE_ENABLE**—packet tracing mode. This parameter can be used for debugging purposes. Set this parameter to 1 to enable packet tracing, or 0 to disable packet tracing.
   - **DOT1X_SUPP_OPERATING_MODE**—mode of operation, wireless (0) or wired (1). Selecting wireless mode will bind the Wind River Wireless Ethernet Driver to the 802.1X interface, and requires the Wind River Wireless Ethernet Driver to be installed, compiled, and configured.

- **DOT1X_SUPP_PASSWORD**—the password to logon with through the 802.1X authenticator.
- **DOT1X_SUPP_USER_NAME**—the name of the user to logon with through the 802.1X authenticator.

3. Include and configure the desired EAP types.

   Under
   802.1X supplicant > EAP (Extensible Authentication Protocol) support, include the EAP types you require for your project:

   - **EAP-LEAP support**
   - **EAP-MD5 support**
   - **EAP-TLS support**
   - **EAP-TTLS support**
   - **EAP-PEAP support**
   - **EAP-MSCHAP-V2 support**

   Depending on which EAP types you include, the **Root certificate support** and/or **Client certificate support** components may be automatically included. These components contain additional configuration for use with digital certificates.

4. Set the optional parameters:

   If **Root certificate support** is included, set the following parameters:

   - **DOT1X_OPENSSLDIR**—root location of the certificates to be used by OpenSSL.
   - **DOT1X_RANDOM_DATA_FILE**—random seed file name.
   - **DOT1X_ROOT_CERTIFICATE**—name of CA (certificate authority) root certificate.
   - **DOT1X_SUPP_TIME_SEC**—seconds after the minute (0-59).
   - **DOT1X_SUPP_TIME_MIN**—minutes after the hour (0-59).
   - **DOT1X_SUPP_TIME_HOUR**—hours after midnight (0-23).
   - **DOT1X_SUPP_TIME_MDAY**—day of the month (1-31).
   - **DOT1X_SUPP_TIME_MON**—months since January (0-11).
   - **DOT1X_SUPP_TIME_YEAR**—current year (4 digit year).
   - **DOT1X_SUPP_TIME_WDAY**—days since Sunday (0-6).

If **Client certificate support** is included, set the following parameters:

- **DOT1X_CLIENT_CERTIFICATE**—location of the client certificate.
- **DOT1X_CLIENT_KEY_FILE**—location of the client private key file.
- **DOT1X_CLIENT_KEY_PASSWORD**—client key password.

Wireless Security Components: Command-Line Configuration

The Wireless Security components can be configured into a VxWorks image using a command-line build. For complete command-line configuration instructions, see *Wind River Wireless Security for VxWorks 6 User's Guide*.

4.7 Configuring Management Components

Wind River Platforms *management* components include:

- Wind River CLI, Web, MIBway
- Wind River SNMP

4.7.1 Wind River CLI, Web, MIBway

Wind River CLI, Web, MIBway cannot be configured into a VxWorks image project in the same way as other Platform components. To add the CLI, Web Server, and MIBway components to your VxWorks system, you must create a **.rcp** project using the Wind River Management Integration Tool. For information about configuring an **.rcp** project, see *Wind River CLI, Web, MIBway for VxWorks 6 Programmer's Guide*.

After you have completed the configuration of your CLI or Web Server project (**.rcp**), use the following steps to create a Workbench project (**.wrproject**) based on your **.rcp** project.

1. With your active **.rcp** project open in the Management Integration Tool, select **Build > Workbench Compatible Project**.

   This generates a **.wpj** project you can then use to create a Workbench project (**.wrproject**).

2. From Workbench, select **File > Import**.

   If you have not yet created a workspace, create one now. For more information about workspaces, see *Wind River Workbench User's Guide*.

3. Select **Existing Tornado 2.x Downloadable Project into Workspace**, then click **Next**.

4. Navigate to the location of your generated **.wpj** project and select one of the following project types:

   - **Downloadable Kernel Module project**
   - **Real Time Process project**

   Click **Finish**.

   For more information about project types, see *Wind River Workbench User's Guide*.

WARNING: Do not delete your **.rcp** project directories after importing to a Workbench project. The Workbench project import copies source files to your workspace, but header files (including the headers generated by the Management Integration Tool) remain in their original locations. If you delete your **.rcp** project directory, you will need to regenerate the header files to build your Workbench project.

5. Set the active build spec.

   a. Select the newly imported project in the Workbench Project Navigator.

   b. Select **Project > Set Active Build Spec**. Choose the same **CPU** and **TOOL** combination you selected when you generated the **.wpj** project with the Management Integration Tool. Click **OK**.

NOTE: You can change the default build specification in Workbench so that you do not need to set the active build spec each time you import a project. To set a default build spec, select **Window > Preferences > Build Properties**, then edit the default values for each type of Workbench project.

6. Link SNMP Libraries for MIBway RTP projects.

   This step is only required if you are using MIBway and building or importing your project as an RTP.

   a. Select **Project > Properties > Build Tools > Linker**.

b. In the field labeled **Command**, append to following to the existing command:
```
-lepcommon -lepdes -lsnmp -lepcommon
```

NOTE: The command argument **-lepcommon** is intentionally repeated

c. Click **OK**.

Wind River CLI, Web, MIBway: Command-Line Configuration

A subset of the code-generation functions of the Management Integration Tool can be run from the command line on any supported host. However, at some point you must create and configure a project using the Management Integration Tool on a supported Windows host. For more information about command-line builds, see *Wind River CLI, Web, MIBway for VxWorks 6 Programmer's Guide* and your release notes.

4.7.2 Wind River SNMP

This section includes basic instructions for including the SNMP agent components in a VxWorks image project. The SNMP agent can be configured in many different ways, some of which require that you re-compile the source. For more information about additional configuration options, including SNMPv3 and AgentX, see *Wind River SNMP for VxWorks 6 Programmer's Guide*.

For more information about SNMP compile-time options, see *3.2.3 Management Component and Feature Descriptions*, p.35.

1. Right-click **Network Components > Network Protocol Components > Network Components for MIB2 > WindManage SNMP Libraries** and select **Include**.
2. Include the required SNMP agent components:
   - **WindManage SNMP Core Library**
3. Include the optional SNMP agent components:
   - **WindManage SNMP V3 Core Support Component**
   - **WindManage SNMP Agentx Common Core**
   - **WindManage SNMP AgentX Master Agent Core**

4. Include optional MIBs:

   When you select the **WindManage SNMP Core Library** component, the standard MIBs for the network stack variant you have compiled (IPv4 or IPv4/IPv6) are automatically included. For more information about these components and other standard MIBs, see the *Wind River Network Stack for VxWorks 6 Programmer's Guide*.

4.8 Configuring User Interface Components

Wind River Platforms *user interface* components include:

- Wind River Media Library

4.8.1 Wind River Media Library

Wind River Media Library provides considerable configuration flexibility. You can build Media Library components as kernel components, RTP projects, or a combination of both. The project creation and build process varies depending on how you choose to configure the Media Library components. For complete project build and configuration instructions, see the *Wind River Media Library for VxWorks 6 SDK Programmer's Guide* and the *Wind River Media Library for VxWorks 6 DDK Programmer's Guide.*

4.9 Configuring Bridging and Routing Components

Wind River Platforms *bridging and routing* components include:

- Wind River Learning Bridge
- Wind River OSPFv2
- Wind River OSPFv3

4.9.1 Wind River Learning Bridge

This section includes basic instructions for including learning bridge components in a VxWorks image project. The learning bridge components require that your BSP support at least two network interfaces. Some BSPs may need to be modified to support additional interfaces. For more information about adding interfaces, see *Wind River Learning Bridge for VxWorks 6 Programmer's Guide*.

To include learning bridge components:

1. Right-click **Network Components > Network Layer 2 Components > Bridge Components** and select **Include**.
2. Select **Learning Bridge**.
3. Set the learning bridge configuration parameters as follows:

CAUTION: You must set these configuration parameters. If you do not, the bridge startup routine will fail.

BRIDGE_IP_ADDR
This is the IP address of the bridge. The type of this parameter is string. Specify a valid IP address enclosed within double-quotes; for example, **"192.168.1.1"**.

BRIDGE_IP_MASK
This is the IP mask of the bridge. The type of this parameter is integer. Specify a valid IP mask; for example, **0xFFFFFF00**.

BRIDGE_PORTS
This is a list of interfaces that are to be added as bridge ports. The type of this parameter is string. Each interface specified must include the device name and unit number. Specify a list of comma-separated interfaces enclosed within double-quotes; for example, **"fei,1,fei,2"**.

NOTE: The IPv6 parameters are optional. If you are not using IPv6, you do not need to set them.

BRIDGE_IPV6_ADDR
This is the IPv6 address of the bridge. The type of this parameter is string. Specify a valid IPv6 address enclosed within double-quotes; for example, **"2001:DB8:1234:100::1"**.

BRIDGE_IPV6_PREFIXLEN
This is the IPv6 prefix length. The type of this parameter is integer. Specify a valid IPv6 prefix length; for example, **64**.

4. Increase the **IP_MAX_UNITS** parameter of the **network init** component, found in the Kernel Editor under
Network Components > Network Core Components > network initialization. **IP_MAX_UNITS** specifies the maximum number of network interfaces supported by the network stack. Add **1** (one) for the learning bridge.

Learning Bridge Components: Command-Line Configuration

The learning bridge components can be configured into a VxWorks image using a command-line build. For complete command-line configuration instructions, see *Wind River Learning Bridge for VxWorks 6 User's Guide.*

4.9.2 Wind River OSPF

This section includes basic instructions for including the OSPFv2 components in a VxWorks image project. Wind River provides both OSPFv2 and OSPFv3 functionality. The OSPF components can be configured in many different ways, some of which require that you re-compile the source. For more information about configuration, initialization, and management options, see *Wind River OSPFv2 for VxWorks 6 Programmer's Guide* and *Wind River OSPFv3 for VxWorks 6 Programmer's Guide*.

For information about compile-time features, see *Wind River OSPF*, p.36.

Wind River OSPFv2

To include the OSPFv2 components:

1. Right-click **Network Components > Network Applications** and select **Include**.
2. Select **OSPF.**
3. If the network stack is configured for virtual stacks, you can specify the virtual stack that OSPF runs in.
   a. Select **OSPF**.

      The properties box, below the component tree, lists the virtual stack **OSPF_VS_ID** parameter.

   b. Set **OSPF_VS_ID** to either **0** (the default) or **1**.

Wind River OSPFv3

To include the OSPFv3 components, right-click **Network Components > Network Applications > OSPFV3** and select **Include**.

OSPF Components: Command-Line Configuration

The OSPF components can be configured into a VxWorks image using a command-line build. For complete command-line configuration instructions, see *Wind River OSPFv2 for VxWorks 6 Programmer's Guide* and *Wind River OSPFv3 for VxWorks 6 Programmer's Guide*.

4.10 Configuring Web Services Components

Wind River Platforms *Web services* components include:

- Wind River Web Services

4.10.1 Wind River Web Services

Wind River Web Services provides a set of development tools and run-time libraries for creating SOAP and XML applications. You can build these XML libraries in several different ways depending on your project requirements. For library build and component configuration instructions, see *Wind River Web Services for VxWorks 6 Programmer's Guide*.

5 Migration

5.1 Platform Migration

This chapter provides information for migrating to Wind River Platforms 3.2.

5.2 Operating System Migration

Migration and backwards compatibility issues for the transition from VxWorks 6.1 to VxWorks 6.2 are covered in detail in *VxWorks Migration Guide, 6.2*. In particular, critical information regarding file system and POSIX changes is available in the migration guide.

NOTE: Wind River recommends that all users migrating from earlier versions of VxWorks consult the migration guide before beginning development.

5.3 Host Tool Migration

This section documents host tool migration issues. Additional information regarding host tool migration issues is available in *Wind River Workbench Migration Guide*.

5.3.1 Wind River Workbench

Detailed information on migrating your existing VxWorks 5.5 applications to Workbench is available in *Wind River Workbench Migration Guide.*

Migrating a VxWorks Image Project created with Workbench 2.2 or 2.3

In the past, the **.wrmakefile** template used to generate the Makefiles for the VIP project contained all functionality on how to build VIP projects.

In Workbench 2.4, the VIP-specific instructions moved to a dedicated **vxWorks.makefile**, which now contains the necessary functionality to build the VIP. The **.wrmakefile** now only covers generic managed build process instructions like recursion and so forth.

Therefore, when migrating existing VIP projects to Workbench 2.4, you must update the makefile template manually as follows:

- If you have purchased and installed Workbench 2.4 and want to continue using VxWorks 6.1, copy over the newly installed (version 2.4) **.wrmakefile** and **vxWorks.makefile** to your existing VIP project. This will allow the project to work properly with the new build system. The simplest way to get these

files is to create a new VIP (using the defaults), copy over the two files, and delete the VIP again.

- If you also purchased and installed VxWorks 6.2, then you must not only copy over the above two files but also run **tcMigrate** to migrate your VIP project from VxWorks 6.1 to 6.2.

  For more information on migrating to a new version of VxWorks, see the **tcMigrate** help entry (by typing **tcMigrate** into the help system **Search** field) and *Wind River Workbench Migration Guide: Workbench Projects*.

NOTE: If you made any manual modifications to your previous **.wrmakefile** file, you must manually migrate those to the new version of the file.

Using Workbench 2.4 for VxWorks 6.1 Development

If you want to use Workbench 2.4 for VxWorks 6.1 development, you must install Workbench 2.4 in the same root directory as the existing VxWorks 6.1 installation. You must also ensure that the **vxworks-6.1** package inherits the **workbench-2.4** environment and not the existing **workbench-2.3** environment. This is accomplished by editing the *installDir*/**install.properties** file after installation. (SPR 107929)

Replace the following line in the **install.properties** file:

```
vxworks61.eval.05=require workbench-2.3
```

with:

```
vxworks61.eval.05=require workbench-2.4
```

You can verify your setup by opening a VxWorks 6.1 shell—no error should be reported:

On Windows:

```
installDir\wrenv.exe -p vxworks-6.1
```

On Linux and Solaris:

```
installDir/wrenv.sh -p vxworks-6.1
```

Using Workbench 2.4 with Older VxWorks Versions

When running Workbench 2.4 with older versions of VxWorks, you may see a dialog reporting that a package named **workbench-2.3** is not found.

To correct this, edit the *installDir*/**install.properties** file and add the following lines to the end of the file:

```
workbench23.name=workbench-2.3
workbench23.version=2.3
workbench23.type=workbench
workbench23.label=Wind River Workbench 2.3
```

5.3.2 Wind River GNU Compiler

The Wind River GNU Compiler 3.3 remains the same for this Platform release, no migration is required.

5.3.3 Wind River ScopeTools

Wind River ScopeTools 5.4 are fully compatible with the previous release, Wind River ScopeTools 5.3.

5.4 Connectivity Component Migration

This section documents migration information for connectivity components. For more information on changes in these components, see the Platform release notes or the specific component documentation.

5.4.1 Wind River CAN

Wind River CAN 1.5.1 is fully compatible with the previous release, Wind River CAN 1.5.

5.4.2 Wind River DCOM

Wind River DCOM 2.3.2 is fully compatible with the previous release, Wind River DCOM 2.3.

5.4.3 Wind River Network Stack

Wind River Network Stack 3.0 is compatible with the previous release, Wind River Network Stack 2.1. However, the IPv6 socket options have changed to comply with updates to the relevant RFCs. For more information, see the release notes.

5.4.4 Wind River OPC

Wind River OPC 3.1.1 is fully compatible with the previous release, Wind River OPC 3.1.

5.4.5 Wind River PPP

Wind River PPP 2.2.1 is fully compatible with the previous release, Wind River PPP 2.2.

5.4.6 Wind River TIPC

Wind River TIPC 1.2 is fully compatible with the previous release, Wind River TIPC 1.1.

5.4.7 Wind River USB

For USB host stack migration from USB 1.x to Wind River USB host stack 2.2 and later, some BSP changes may be required for BSPs that provide non-PCI support. See the *USB Host Stack Programmer's Guide, 2.2* and **usrUsbPciInit.c** file for detailed information on the non-PCI implementation.

5.4.8 Wind River Wireless Ethernet Driver

The Wind River Wireless Ethernet Driver has the following migration issues:

BSP Integration

You may need to perform *manual* BSP modifications—that is updates to **config.h**, **configNet.h**, and **sysLib.c**—for VxWorks to recognize your wireless hardware, if

these changes are not already in the BSP. For more information, see the *Wind River Wireless Ethernet Driver for VxWorks User's Guide*.

Multiple Driver Use

Although the driver maintains **ioctl( )** backwards compatibility for existing applications that use the 802.11 driver, legacy public hardware-specific routines—that is, routines beginning with *intPrism*—are not supported in this driver.

5.5 Security Component Migration

This section documents migration information for security components. For more information on changes in these components, see the Platform release notes or the specific component documentation.

5.5.1 Wind River IPsec and IKE

Wind River IPsec and IKE 3.2 has the following migration issue:

Perfect Forward Secrecy Option

The Perfect Forward Secrecy (PFS) option in **ikeAddPeerAuth( )**, which was deprecated in the previous release of Wind River IPsec and IKE, has now been removed, since PFS is now enabled by specifying a Diffie-Hellman group.

Specifying the **KEYPFS**, **PFS**, or **NOPFS** keyword in the parameter string of **ikeAddPeerAuth( )** now causes a syntax error and a return value of -1. To update an existing parameter string, remove the **KEYPFS**, **PFS**, or **NOPFS** keyword and its trailing comma.

ikeShowPeerAuth( ) no longer shows the status of the PFS option.

WARNING: Your Wind River IPsec and IKE configuration will not work with this release until you remove the **KEYPFS**, **PFS**, or **NOPFS** keyword from the parameter string of **ikeAddPeerAuth( )**.

5.5.2 Wind River Firewall

Wind River Firewall 2.1 has the following migration issue:

Changes in Rate Limiting

Wind River Firewall 2.1 extends rate limiting to add host tracking, and adds two new parameters to **fwRuleFieldSet( )** for the field **FW_RATE_LIMIT**.

For example, to set the rate limit condition of > 500 packets/minute using Wind River Firewall 2.0, you called the **fwRuleFieldSet( )** routine with the following arguments:

```
fwRuleFieldSet (id, FW_FIELD_RATELIMIT, FW_GT_OP, 500, 60);
```

Using Wind River Firewall 2.1, to set the same rate limit condition, call the **fwRuleFieldSet( )** routine as follows:

```
fwRuleFieldSet (id, FW_FIELD_RATELIMIT, FW_GT_OP, 500, 60,
                FW_ALL_TRK_OFF, 0);
```

5.5.3 Wind River NAT

Wind River NAT 2.1 is fully compatible with the previous release, Wind River NAT 2.0.

5.5.4 Wind River RADIUS Client

The Wind River RADIUS Client 1.3 requires no migration from Wind River RADIUS Client 1.2 if you are using IPv4. Changes are required in order to use IPv6. For more information, see *Wind River RADIUS Client for VxWorks 6 User's Guide, 1.3*.

5.5.5 Wind River SSL

Wind River SSL 1.1 is fully compatible with the previous release, Wind River SSL 1.0.

5.5.6 Wind River Security Libraries

Wind River Security Libraries 1.0 is fully compatible with the previous release, Wind River Security Libraries 1.1.

5.5.7 Wind River Wireless Security

This version of Wind River Wireless Security can be easily installed over the last released version, that is 2.0. For more information on specific changes to the API reference material, see your Platform release notes.

5.6 Management Component Migration

This section documents migration information for management components. For more information on changes in these components, see the Platform release notes or the specific component documentation.

5.6.1 Wind River SNMP

Wind River SNMP 10.0.2 is fully compatible with the previous release, Wind River SNMP 10.0.

5.6.2 Wind River CLI, Web, MIBway

Wind River CLI, Web, MIBway 4.4.2 is fully compatible with the previous release, Wind River CLI, Web, MIBway 4.4.

5.7 User Interface Component Migration

This section documents migration information for user interface components. For more information on changes in this component, see the Platform release notes or the specific component documentation.

5.7.1 Wind River Media Library

Wind River Media Library 4.0 is fully compatible with the previous release, Wind River Media Library 4.1.

5.8 Bridging and Routing Component Migration

This section documents migration information for bridging and routing components. For more information on changes in these components, see the Platform release notes or the specific component documentation.

5.8.1 Wind River Learning Bridge

Wind River Learning Bridge 1.3.1 is fully compatible with the previous release, Wind River Learning Bridge 1.3.

5.8.2 Wind River OSPFv2

Wind River OSPFv2 3.1.1 is fully compatible with the previous release, Wind River OSPFv2 3.1.

5.8.3 Wind River OSPFv3

Wind River OSPFv3 3.1.1 is fully compatible with the previous release, Wind River OSPFv3 3.1.

5.9 Web Services Component Migration

This section documents migration information for the Web services component. For more information on changes in this component, see the Platform release notes or the specific component documentation.

5.9.1 Wind River Web Services

Wind River Web Services has the following migration issues:

XML Pull Parser Structure Field Access

In Wind River Web Services 1.3, the individual fields of the XPP structure are no longer visible to the application programmer. Previously, the only way that you could access the error string was by pointing directly to the **errorString** field in the **XPP** structure. Change your application code to access the **errorString** through the **xppGetErrorString( )** routine.

In previous versions of Wind River Web Services, you accessed the XML pull parser error string using a method such as:

```
errStr = pXpp->errorString;
```

In Wind River Web Services 1.3, you access the XPP error string through a call to the **xppGetErrorString( )** routine, as in the following code fragment:

```
errStr = xppGetErrorString(pXpp);
```

SAX Header File

Wind River Web Services 1.3 no longer includes the SAX parser header file, which was formerly located at:

installDir/**components/webservices-1.2/tutorials/xml/examples/xpp/sax.h**

You must remove any references to this file from your application code.

A WDB over TIPC

A.1 Introduction

TIPC communication does not require any of the services provided by the UDP and TCP/IP protocols. Because of this, many nodes on a TIPC network (for example, VxWorks targets compiled with the **TIPC network stack** build option) lack the facilities to communicate directly with a development host outside of the TIPC network. In order to facilitate debugging and other host tool communication between a minimal TIPC node and Wind River Workbench, Wind River provides the WDB over TIPC feature for both VxWorks and Linux target operating systems.

This appendix provides a high level overview of the setup, configuration, and target connection process for WDB over TIPC for both VxWorks- and Linux-based systems. In particular, because a configuration where the TIPC network includes both VxWorks and Linux targets (for example, where a Linux target is the gateway and a VxWorks target compiled with the **TIPC network stack** build option is a TIPC node) is supported, this appendix illustrates how this process is parallel for

VxWorks and Linux targets. This appendix also includes references to additional documentation where you can go to find further information for both target types.

NOTE: This appendix does not provided detailed information for proper configuration and setup of WDB over TIPC on either VxWorks or Linux targets. You *must* review the documentation referenced in this appendix before configuring and running WDB over TIPC.

The WDB Target Agent

The WDB target agent is a run-time facility that allows you to connect Workbench or other host tools to a target using a target server. For more information on WDB, see *VxWorks Kernel Programmer's Guide: Target Tools* (VxWorks targets) or *Wind River Workbench User's Guide, Linux Edition: TIPC Configuration* (Linux targets).

TIPC

TIPC is a transparent inter-process communication (TIPC) infrastructure used for inter-node (cluster) communication. For more information on TIPC, see the open source TIPC Web site:

http://tipc.sourceforge.net/

For information on configuring and using Wind River TIPC with VxWorks, see *Wind River TIPC for VxWorks 6 Programmer's Guide*. For information on using TIPC with Linux targets, see *Wind River Workbench User's Guide, Linux Edition: TIPC Configuration*.

A.2 Setup Requirements

In order to use WDB over a TIPC network, you must set up your TIPC network according to some basic guidelines.

Setting Up Your TIPC Network

The following minimum setup for your TIPC network is required:

- You must have a minimum of two targets on your TIPC network and both targets must be located in the same zone and cluster.

- One of the targets on your TIPC network must act as a gateway. This target must have access to both the TIPC network and an IP network that is accessible to your host system.

This setup allows you to connect to the remaining target (or targets) on the TIPC network using WDB over TIPC.

A.3 Target Configuration

This section provides an overview of how to configure your gateway and TIPC nodes to use WDB over TIPC. For VxWorks targets, specific information regarding how to configure your VxWorks image is available in the *VxWorks Kernel Programmer's Guide: Target Tools* and the *Wind River TIPC for VxWorks 6 Programmer's Guide*. For Linux targets, see *Wind River Workbench User's Guide, Linux Edition: TIPC Configuration*.

Configuring the Gateway

The target you plan to use as a gateway to your TIPC network must include TIPC support as well as support for the WDB proxy agent. For VxWorks targets, you must include the **INCLUDE_WDB_PROXY** and **INCLUDE_WDB_PROXY_TIPC** components. For Linux targets, be sure you have installed the TIPC kernel module and use the **wrproxy** command to set up the WDB proxy agent.

Configuring the TIPC Nodes

To configure the target you want to connect to using WDB over TIPC, you must include and configure TIPC support on the target. You must also include a WDB agent and a WDB agent TIPC communication link for the target. For VxWorks targets, you can include the WDB agent TIPC communication link by including the **INCLUDE_WDB_COMM_TIPC** component in your VxWorks image or by setting **WDB_COMM_TYPE** to **WDB_COMM_TIPC** in your BSP **config.h** file. On Linux targets, launch the **usermode-agent** on the target you want to connect to.

NOTE: For VxWorks targets, TIPC nodes requiring WDB over TIPC support for host communication are configured with the **TIPC network stack** build option. For more information, see *Wind River TIPC for VxWorks 6 Programmer's Guide: Building VxWorks to Include Wind River TIPC*.

In addition to the above configuration, you must configure the WDB TIPC port type and the WDB TIPC port instance that you plan to use for WDB communication. You can accept the default values for these parameters or you can change the settings as necessary. For VxWorks targets, change the settings by altering the **WDB_TIPC_PORT_TYPE** and the **WDB_TIPC_PORT_INSTANCE** parameters (for more information, see *VxWorks Kernel Programmer's Guide: Target Tools*). For Linux targets, you can specify the WDB TIPC port type and port instance on the command line when starting the debug agent (for more information, see *Wind River Workbench User's Guide, Linux Edition*).

A.4 Establishing a Target Server Connection

To establish a target server connection to a target on your TIPC network, do the following:

1. Boot the targets on your network. This includes the target that will act as the TIPC gateway as well as the target you wish to connect to.
2. Start the target server on your host using the following command:

   -> **tgtsvr -V -B wdbproxy -tipc -tgt** *targetTipcAddr* **-tipcpt** *tipcPortType* **-tipcpi** *tipcPortInstance wdbProxyIPAddr*

   where *targetTipcAddr* is the TIPC address of the target you wish to connect to, *tipcPortType* is the TIPC port type used for the connection, *tipcPortInstance* is the TIPC port instance used for the WDB connection, and *wdbProxyIPAddr* is the IP address (or target name) of the target that is running the WDB proxy agent.

For more information on establishing a target server connection to a target on your TIPC network, see *Wind River Workbench User's Guide, VxWorks Edition: New Target Server Connections* (VxWorks targets) or *Wind River Workbench User's Guide, Linux Edition: TIPC Configuration* (Linux targets).

B

Removing Run-time Libraries

B.1 Introduction

By default, both the Wind River Compiler and GNU GCC run-time libraries are included in the VxWorks image. If you want to remove either the Wind River Compiler or GNU GCC libraries, follow these instructions.

B.2 Removing Run-time Libraries from VxWorks Images

By default, VxWorks includes both **libc** and GNU **libgcc**, which are provided with the **INCLUDE_ALL_INTRINSICS** component. To exclude one library or the other, reconfigure the kernel with either **INCLUDE_DIAB_INTRINSICS** or **INCLUDE_GNU_INTRINSICS**, respectively.

NOTE: The **libc** and **libgcc** libraries are available in the kernel to enable dynamic downloading and running of kernel object modules.

Index

B

C

D

E

F

G

H

I

K

M

N

O

P

R

S

T

U

V

W

X