Spectra ORB
C and C++ Editions
Product and Installation Guide

Version 2.0

Guide edition: 04 (24/04/13)

Table of Contents

1 Preface

1.1 About the Product and Installation Guide

The Product and Installation Guide is included with the Spectra ORB SDR (formally e*ORB) Documentation Set. This guide is the starting point for anyone using, developing or running applications with Spectra OR C and C++ Editions. The Product and Installation Guide contains:

- general information about Spectra ORB
- a list of documents and how to use them
- initial installation and configuration information
- details of using the product on specific supported platforms
- details of where additional information can be found, such as the Spectra FAQs, Knowledge Base, bug reports, etc
- a Bibliography listing background information, recommended texts and reference material which users and developers may find useful

1.1.1 Intended Audience

The Product and Installation Guide is intended to be used by anyone who wishes to use the Spectra ORB product.

1.1.2 Conventions

The conventions listed below are used to guide and assist the reader in understanding the Guide.

!	Item of special significance or where caution needs to be taken
i	Item contains helpful hint or special information
WIN	Information applies to Windows (*e.g.* XP, Vista, Windows 7) only
UNIX	Information applies to Unix based systems (*e.g.* Solaris) only
C	C language specific
C++	C++ language specific
Java	Java language specific

Hypertext links are shown as *blue italic underlined*.

On-Line (PDF) versions of this document: Items shown as cross-references to other parts of the document, *e.g.* *Contacts* on page 7, behave as hypertext links: users can jump to that section of the document by clicking on the cross-reference.

```
% Commands or input which the user enters on the command
line of their computer  terminal
```

`Courier`, **`Courier Bold`**, or ***`Courier Italic`*** fonts indicate programming code and file names.

Extended code fragments are shown in shaded boxes:

```
NameComponent newName[] = new NameComponent[1];

// set id field to “example” and
// kind field to an empty string

newName[0] = new NameComponent (“example”, “”);
rootContext.bind (newName, demoObject);
```

Italics and ***Italic Bold*** indicate new terms, or emphasise an item.

Sans-serif Bold indicates user related actions, *e.g.* **File > Save** (a sequence of selections from menus, or buttons or check-boxes).

Step 1: One of several steps required to complete a task.

1.2 Contacts

PrismTech can be contacted at the following contact points.

Corporate Headquarters

PrismTech Corporation
400 TradeCenter
Suite 5900
Woburn, MA
01801
USA

Tel: +1 781 569 5819

European Head Office

PrismTech Limited
PrismTech House
5th Avenue Business Park
Gateshead
NE11 0NG
UK

Tel: +44 (0)191 497 9900

Fax: +44 (0)191 497 9901

Web: *http://www.prismtech.com*

Technical questions: *crc@prismtech.com* (Customer Response Center)

Sales enquiries: *sales@prismtech.com*

2 Introduction

This guide covers the basic installation and configuration of Spectra ORB C and C++ Editions. Both ORBs share the same installation hierarchy and configuration mechanism. A particular distribution may contain one or both of these ORBs in a single distribution package. Some common components, such as transport libraries are shared between both ORB implementations. A distribution package may also contain libraries and binaries for multiple target platforms, typically where cross-developing for an embedded target such as VxWorks, LynxOS, QNX or Integrity.

3 About Spectra ORB

3.1 What is Supported

Spectra ORB is a suite of Common Object Request Broker Architecture (CORBA) and Software Communications Architecture (SCA) compliant, high performance, low footprint, ORB and Common Object Services (COS), specifically profiled and optimized for use in Software Defined Radio (SDR) and embedded applications.

Spectra ORB v2 provides a pluggable set of libraries that can be used to configure the product to support the following CORBA profiles:

- Minimum CORBA Profile (both C and C++ ORBs)
- CORBA/e Compact Profile (both C and C++ ORBs)
- CORBA/e Micro Profile (C ORB only)
- Software Communication Architecture v4.0 Full Profile (both C and C++ ORBs)
- Software Communication Architecture v4.0 Lightweight Profile (C ORB only)

The Spectra ORB C and C_+Editions also include a suite of supporting COS Services, specifically:

- Full Naming Service (C and C++)
- Lightweight Naming Service (C and C++)
- Lightweight Event Service (C and C++)
- Lightweight Log Service (C and C++)

Spectra ORB is the market-leading integration infrastructure software product for embedded and real-time applications. It is the only product on the market that has been successfully ported to run not only on General Purpose Processors (GPP), but also on specialised radio hardware devices such as Digital Signal Processors (DSP) and Field Programmable Gate Arrays (FPGA).

This enables waveform functionality to be provided by a single, unified, SCA-compliant solution which

- can be provided across different hardware platforms and processor sets
- can be highly optimized
- is easily configurable
- can provide the high performance and small memory footprint required by SDR and other embedded operating environments

Spectra ORB implements a number of lightweight Profiles based on the Object Management Group’s (OMG) CORBA specification and can be used for C and C++ based SDR and embedded application development. In addition to complying with the OMG's minimum CORBA and CORBA/e specifications, Spectra ORB also satisfies the CORBA requirements of the Joint Tactical Radio System's (JTRS) Software Communication Architecture (SCA) standards (both v2.2.2 and v4.0)

The C++ Edition offers developers all of the advantages of object-oriented programming, combined with high performance and low footprint. The C Edition offers minimum footprint and is the first ORB that can be hosted on GPP, DSP and FPGA soft cores.

Specifically Spectra ORB supports provides support for CORBA APIs and features defined in the following standards:

- Minimum CORBA Specification version 1.0: OMG Document formal/02-08-01, August 2002
- Common Object Request Broker Architecture for Embedded (CORBA/e) Specification Version 1.0: OMG Document Number: formal/2008-11-06
- Software Communications Architecture Specification – Joint Program Executive Office (JPEO) Joint Tactical Radio System (JTRS)– SCA v2.2.2 – FINAL / 15 May 2006
- Software Communications Architecture Specification V4.0, 28th February 2012, - Appendix E: Platform Specific Model (PSM) – Common Object Request Broker Architecture (CORBA)
- Interoperable Naming Service Specification: OMG Document formal/00-11-01
- Lightweight Log Service Specification: OMG Document formal/05-02-02: v1.1
- Event Service Specification: OMG Document formal/05-02-02: v1.1
- Real-time CORBA Specification, January 2005 Version 1.2 formal/05-01-04
- C++ Language Mapping Specification, July 2012, Version 1.3, formal/2012-07-02
- C Language Mapping Specification, June 1999, Version 1.0, formal/1999-07-35

3.2 Product Details

3.2.1 Key Features

- *Micro ORB kernel*
- *IDL to C++ or C compiler*
- *GIOP 1.2 support (with the exception of bi-directional support)*
- *Pluggable Portable Object Adaptor (POA) architecture* containing an extensible and highly scalable plug-in POA architecture (including child POAs)
- *Extensible Transport Framework* – providing multi-transport plug-in support, TCP transport by default; provides users with the ability to develop additional custom transports as required (e.g. UDP, RapidIO, Compact PCI, other..)
- *Native and portable exceptions*
- *Multi-thread safe*
- *User configurable server-side threading* – Thread-Per-Request and Thread-Per-Connection models supported
- *Pluggable Any data type support*
- *Request time-outs*
- *Pluggable Real-time CORBA support*
- *Suite of Lightweight Common Object Services* – Naming Service (both Full and Lightweight), Event Service, Log Service

3.2.2 Platforms

Spectra ORB has been ported to a wide variety of platforms, including:

- Linux
- Solaris
- AIX
- HP-UX
- BSD
- Integrity
- VxWorks
- LynxOS
- OSE
- QNX
- VOS (Stratus)
- DSP/BIOS (Texas Instruments)
- VDK (Analog Devices)
- Windows, including WinCE

The ORB has been designed to be portable to any operating system and runs on a variety of target processors (x86, Itanium, ARM, PowerPC, SPARC, SuperH, DSP). To request the ORB on an operating system not listed above or a specific hardware platform, please contact PrismTech.

4 Installation

This chapter describes how to install Spectra ORB. Please follow the procedures carefully.

The installation files can be downloaded from the PrismTech Web site *http://www.prismtech.com*.

An installation file is typically simply a compressed directory archive. For UNIX platforms this can be a `tar.gz` file (a tar file compressed with GNU `gzip`) or a `tar.Z` file (a tar file compressed with the native compress). For Windows platforms this is a zip file.

4.1 Installation Directories

The ORB distribution is installed under the `eorb` directory as follows:

Directory	Description
`bin`	ORB executables. Executables for each supported configuration are located in the subdirectory corresponding to the EORBENV environment variable.
`docs`	The HTML and PDF documentation sets for the C and C++ ORBs.
`etc`	Default location for miscellaneous configuration files, such as the licence file.
`examples`	ORB example applications. C and C++ examples are in separate sub directories.
`include`	ORB header and IDL files.
`lib`	Pre-built libraries for the target environment. Also, Java jar files required by the IDL compiler. Libraries for each supported configuration are located in the subdirectory corresponding to the EORBENV environment variable. C ORB libraries have the **`ec`**_ prefix and C++ ORB libraries the **`e`**_ prefix.
`make`	Configuration files used by the Spectra ORB *make* system to build the code examples. Note: customers should **not** rely on these make files for their own build processes as they may change between releases.

4.2 Environment Variables

The following environment variables should be defined:

Variable	Usage
`EORBENV`	A configuration name representing the target system.
`EORBHOME`	The directory where the ORB is installed.

See the html release notes for the latest set of supported configurations.

4.3 System Variables

System variables need to be modified to include the installation `bin` directory in the executable search path, unless executables are to be invoked by a fully-scoped path name. Similarly the ORB

shared libraries in the `lib` directory must be included in the shared library search path. The following system variables may need to be modified:

Variable	Usage
`PATH`	The path used to locate executables. Should include `<EORBHOME>/bin/<EORBENV>` and on Windows should also include the path to shared libraries in `<EORBHOME>/lib/<EORBENV>`.
`LD_LIBRARY_PATH`	The path used to locate shared libraries. Should include `<EORBHOME>/lib/<EORBENV>`

The `LD_LIBRARY_PATH` is not used on all systems (such as Windows) and may not be required if static rather than shared libraries are used. Also some systems can use other mechanisms, such as configuration files, to set the shared library search path. For cross-development environments, such as compiling for VxWorks on a Windows host, the `bin` directory of the host also needs to be on your `PATH` so the IDL compiler and other utilities can be executed.

4.4 Installation Procedure

4.4.1 General

All installed files are placed in the installation directory specified during installation - no files are stored in any of the UNIX or Windows system directories. The software can be completely removed from a system by simply deleting the installation directory.

4.4.2 Preparation

It is recommended that any existing installation be removed before installing the current version. To support multiple different versions simply install each version into a different directory. Note that by default the distribution installs into an `eorb` subdirectory; this may overwrite an existing installation.

4.4.3 Extract the Distribution

On UNIX systems simply uncompress and de-tar the installation file. For example on Linux:

```
> tar xzf <file>.tar.gz
```

On Windows systems simply double click on the installation file to uncompress or explicitly uncompress using one of a variety of freely available unzip programs.

4.4.4 Install the License File

A valid license file must be placed into the `<EORBHOME>/etc` directory after extracting the distribution. Please note that the product will not run without a valid license file. PrismTech uses RLM for product licensing. Note that licensing is not supported on all systems.

License files are provided by PrismTech for services or products which have been purchased. Contact PrismTech for purchasing details.

Evaluation licenses are provided when an Spectra product is downloaded from the PrismTech Web site *http://www.prismtech.com/*

See Chapter *7* for more details on licensing.

4.4.5 Test the Installation

The installation can be tested by building and running the examples provided in `<EORBHOME>/examples`. Details on how to do this are included in the html documentation.

4.4.6 Removing an Installation

The installation can be completely removed by simply deleting the installation directory after stopping any ORB-based applications or services.

5 General Compilation Issues

5.1 Compilation Flags

The compilation flags used by a particular Spectra ORB build can be found in the `compile_flags.txt` file in the `etc` directory of the installation. Building the code examples will also show how they are compiled when verbose compilation is enabled; for example:

```
gmake VERBOSE=
```

5.2 Linking with Spectra ORB Libraries

Two sets of Spectra ORB libraries are provide with a distribution, one set for debugging purposes and one set for production purposes. The debug libraries are distinguished by having the suffix **_g** on the library name. Note that the debugging libraries contain code with additional assertions and checking enabled. All C library names start with **ec**_ and all C++ libraries with **e**_.

5.3 Operating System Specific Define

A compile time C pre-processor define is used to select the appropriate headers for inclusion. The required define is based on a translation of the `EORBENV` environment variable where dash characters are replaced by underscore characters. For example, an `EORBENV` value of `linux-gcc-x86` would require `linux_gcc_x86` to be defined with an appropriate compilation flag (typically `-D`).

6 System-specific Issues

This section describes the requirements for using, compiling, linking, deploying and running Spectra ORB on specific supported platforms.

i PrismTech is continually adding to the range of supported platforms. Refer to the release notes included with your Spectra ORB distribution or contact PrismTech Support if your platform is not covered in this section.

6.1 Unix

The requirements for using Spectra ORB on Unix platforms are given in this section. Supported platforms include Linux, BSD, Solaris, HPUX, and AIX.

6.1.1 Environment

The required values for the Unix and Spectra ORB environment variables are described below. The common Spectra ORB environment variable settings are:

- *EORBENV* - set its value as shown in Chapter 10, Build Environments
- *EORBHOME* - set its value to the directory where Spectra ORB is installed

There are no other configuration or environment variable requirements for the Unix target platforms.

Set `PATH` to include:

```
$EORBHOME/bin/$EORBENV
```

and `LD_LIBRARY_PATH` to include:

```
$EORBHOME/lib/$EORBENV
```

For Unix systems, libraries are usually shared, although static libraries can also be provided if required.

6.1.2 Compiling

Spectra ORB-based applications are often compiled using the *GNU gcc* compiler (normally provided with a Linux distribution or obtainable from the *GNU GCC* web site at *http://gcc.gnu.org*).

The Spectra ORB compiler options are described in section *5.3, Operating System Specific Define*, on page *15*.

6.1.3 Linking

Spectra ORB-based applications are often linked using the *GNU gcc* compiler (normally provided with a Linux distribution or obtainable from the *GNU GCC* web site at *http://gcc.gnu.org*).

The Spectra ORB-required linking options are described in section *5.2, Linking with Spectra ORB Libraries, on 15.*

6.1.4 Deploying and Running

Server resolution methods for running UNIX-based applications are described below.

6.1.4.1 Resolution Using an IOR File

The server saves its IOR (as a string) to a file when the server is started. Clients open the file, read the IOR as a string, then convert the string to an object reference to the server. Server resolution is handled transparently by the server and client programs.

Step 1: Run the server in a terminal window; for example:

```
% server -ORBRegistry file:
```

Step 2: Run the client in another terminal window; for example:

```
% client -ORBInitRef <name>=file:<file>
```

6.1.4.2 Resolution Using the Server's Printed IOR

By default a server prints the initial reference that the client can use to resolve it.

Step 1: Start the server in a terminal window; for example:

```
% server
```

Step 2: Start the client in another terminal window using the `-ORBInitRef` argument output by the server:

```
% client -ORBInitRef <name>=IOR:...
```

6.2 Integrity

This section describes the requirements for using Spectra ORB on Green Hills Integrity platforms. Spectra ORB supports a variety of Integrity versions on different chip sets (x86, ARM and PowerPC) listed in Section *10.1, Supported native build environments, on page 32*. Contact PrismTech for availability on specific hardware systems.

6.2.1 Environment

Spectra ORB requires *POSIX* enabled; see the Integrity manual '*Libraries and Utilities User's Guide*' v5.0, Chapter 21 (specifically section 21.4.2.1.1, *Setting Up the POSIX System Server*).

For pre-built binaries to run correctly you must build your Integrity kernel with the following extra libraries: `itcpip, posix_sys_server_kernel, ivfs, posix_authclient, ramdisk, ffs` and `ivfsserver` and remove the load library.

Licensing is not supported on Integrity.

The required Spectra ORB environment variable settings are:

- `EORBENV` - set its value as shown in Chapter 10 Build Environments (starting on page 32).
- `EORBHOME` - set its value to the directory where e*ORB is installed

The Integrity-specific environment variable settings are:

- `BSPNAME` - set to the name of the Integrity BSP's main build directory.
- `I_INSTALL_DIR` - set its value to the Integrity host development installation directory.

6.2.2 Compiling

The compiler options should be set as described in Section *5, General Compilation Issues, starting on page 15.*

6.2.3 Linking

The Spectra ORB required linking options are described in Section *5.2, Linking with Spectra ORB Libraries, on page 15*. The Integrity-specific linking options are described below.

6.2.3.1 Integrity Libraries

- `-lposix` - provides POSIX thread and semaphore support.
- `-lposixsca` - provided for SCA compliant applications. This library can be used as an alternative to the `posix` library if required. Refer to Section *5.2, Linking with Spectra ORB Libraries, on page 15*, for the additional steps needed when using the `posixsca` library.

6.2.3.2 Transport Libraries

The following libraries are needed when the Integrity GHnet V2 TCP/IP stack is used:

- `-lnetconfig` - TCP/IP network configuration
- `-lsocket` - Integrity socket library
- `-lnet` - Integrity's network and internet address functionality

Refer to Green Hills documentation for information on the configuration of Integrity kernels with the GHnet V2 TCP/IP stack.

6.2.3.3 System and Board Support Libraries

The location of the system and board support libraries must be provided when linking against these libraries using the `-L` flag, as shown:

- `-L$(I_INSTALL_DIR)/$(BSPNAME)`

6.2.3.4 Directive Files

The `eORB.ld` linker directives file is required. This file is located in the `etc/integrity` sub-directory.

This file specifies a set of default linker directives for Spectra ORB applications. The user may choose to override or provide alternative settings. The approach to including this file varies according to the host environment. On UNIX hosts, this file is automatically used by the supplied example make system. When using the *Multi IDE*, the `eORB.ld` file must be manually added to the project. Refer to the Integrity documentation for information on this process.

i The file must be explicitly added to the project's link line if using a Windows host.

6.2.3.5 Dynamic Download or Monolithic Image

The build requirements for your Spectra ORB-based application depends on whether you are creating a dynamic download image (which can be loaded onto the target board alongside a pre-running Integrity kernel) or whether both kernel and application are combined into a single, monolithic image.

The linking options described below are needed when building a dynamic download.

Integrity's *Integrate* utility is required to create an Integrity application image. The *Integrate* utility can be run using either the Integrate IDE (refer to the Integrity documentation) or from the command line by specifying `-integrate` as a linking option.

The `-dynamic` linking option then specifies that the final application is a dynamic download application. Refer to Green Hills documentation for comprehensive descriptions and detailed information about the image type and creation process.

The requirements specific to Integrity platforms for deploying and running Spectra ORB-based applications are described here.

A downloadable application should be dynamically downloaded to a target server using the *MULTI IDE*. This can be done by:

Step 1: Selecting **Target... > Connect To Target** from the IDE.

Step 2: Select the appropriate target board from the drop down menu and select **Connect.**

Step 3: After the connection is established, select **Target... > Load Module** to load your application onto the target board, for example, `server.mod`.

Step 4: Double-click the *Initial* task for this process to prepare the process for running.

Step 5: Click the **Go** button (the green triangle) to run the executable.

Refer to the Green Hills documentation for additional details on downloading and running applications on Integrity platforms.

6.2.4 Network Configuration for an Integrity ORB

Spectra ORB is configured to use a sockets based TCP/IP stack by default. Consult the Integrity documentation for more information on how to build TCP/IP into an Integrity system.

As described above, servers publish IORs which are used by clients to establish a network connection to that server. The ORB requires that clients and servers must each have an associated IP address before a connection can be established.

Target boards must be configured with unique IP addresses on Integrity platforms that use GHnet V2 TCP/IP. Spectra ORB uses stack specific API calls for determining the IP address and port numbers to be placed in IORs as address information in IIOP profiles.

See the Integrity documentation for information on how to set the host name, IP addresses and other network configuration parameters.

6.2.5 Memory and Resource Configuration

It may be necessary to increase various resource and memory defaults in order to handle many simultaneous ORB clients in a system.

Most of these are hidden from the user and are accounted for in the `eORB.ldlinker` directive file and `eORB.bsp` files located in the `$EORBHOME/etc/integrity` directory residing on the host.

These files are automatically used by the sample make system on UNIX hosts. These files must be explicitly added to projects on Windows hosts or when using the MULTI Builder on UNIX.

One important configuration is the `.heap` setting in the `eORB.ld linker` directive file. This figure determines the heap space available to Spectra ORB applications. The default setting is `0x100000`, but this may need to be increased in order to accommodate a larger number of clients and tasks.

The Integrity TCP/IP stack must also be configured to handle a larger number of ORB clients, which is typical on a production system. The `.heap` section will likely need to be increased to account for this. See the Integrity documentation for more information on how to increase the heap used for TCP/IP.

6.3 VxWorks

This section describes the requirements for using Spectra ORB with VxWorks platforms. Spectra ORB supports a variety of VxWorks versions on different chip sets (x86, ARM, PowerPC and SuperH) listed in Section *10.1, Supported native build environments*, on page 32. Contact PrismTech for availability on specific hardware systems.

Allocating and freeing sockets quickly may make VxWorks run out of resources. The socket, or accept call returning `-1` is the typical symptom of this which can show up as a `CORBA COMM_FAILURE` system exception. The solution to this is to increase the maximum number of file handles, and the space reserved for the TCP stack. This is done from the kernel configuration within *WorkBench*:

Under **Operating System Components > IO System Components > IO System** increase the value of `NUM_FILES`.

The following VxWorks (6) kernel components need to be included to support Spectra ORB:

Component	Required
FOLDER_POSIX	`Yes`
SELECT_PX_SCHED_POLICIES	`Yes`
FOLDER_CPLUS	`Yes if C++ ORB used`
FOLDER_RTP	`Yes if RTPs used`

Licensing is not supported on VxWorks.

6.3.1 Environment

The required Spectra ORB environment variable settings are:

- `EORBENV` - set its value as shown in the `EORBENV` values shown in Section *10.2, Supported cross build environments*, on page 33
- `EORBHOME` - set its value to the directory where Spectra ORB is installed.

There are no required VxWorks specific environment variables for the VxWorks target platforms. See the VxWorks documentation for configuration and build process requirements for your specific host system.

6.3.2 Compiling

The Spectra ORB-related compiler options should be set as shown in Section *5, General*

Compilation Issues, on page *15*.

6.3.3 Linking

The required Spectra ORB linking options are given in Section *5.2, Linking with Spectra ORB Libraries*, on page *15*.

There are no required VxWorks-specific linking options.

6.3.4 Deploying and Running

The requirements specific to VxWorks for deploying and running Spectra ORB-based applications are described here.

A downloadable kernel application should be downloaded to a target server using the *Tornado* module loader when using VxWorks 5.5.1. This can be performed using the *Tornado* shell `ld` command. For example:

```
ld 1,0, "server.out"
```

Alternatively, the **Download** option from the *Tornado* IDE may be used.

After downloading, an application can be started by invoking the appropriate application routine from the *Tornado* shell. Additional arguments may be passed to the routine for use by Spectra ORB, for example:

```
client ("-ORBInitRef", "hello=IOR:0000....")
```

For more details on downloading and running applications please refer to the *Tornado User's Guide.*

6.4 LynxOS

This section describes the requirements for using Spectra ORB on LynxOS platforms.

LynxOS versions 4.0, 4.2 and 5.0 are currently supported. Any platform can be supported for which a BSP is available, although only a subset of hardware may be available for in house testing.

Refer to the LynxOS documentation for information about how to use and develop applications for LynxOS.

Spectra ORB can be built natively on a LynxOS host (typically x86), in which case, LynxOS can be treated as a Unix system. Licensing is not supported on LynxOS.

Note that for C ORB development, a Java runtime is required as the IDL compiler is a Java application. LynxOS distributions do not usually include Java support and so are not recommended as host development systems for the C ORB, although third-party Java support is available.

6.4.1 Environment

The required Spectra ORB-related environment variables are:

- `EORBENV` - set its value as shown below.
- `EORBHOME` - set its value to the directory where Spectra ORB is installed

There are no required LynxOS-specific environment variable for the LynxOS target platforms. LynxOS versions 4.0, 4.2 and 5.0 are currently supported on x86 and PowerPC.

Platform	EORBENV value
LynxOS host and target	`lynxos-gcc-x86`
Windows host LynxOS PowerPC target	`lynxos-gcc-ppc-win32`
Linux host LynxOS PowerPC target	`lynxos-gcc-ppc-linux`
Linux host LynxOS x86 target	`lynxos-gcc-x86`

6.4.2 Compiling

The Spectra ORB-related compiler options should be set as described in Section *5, General Compilation Issues*, on page *15*.

The LynxOS-specific compiler options should be set as shown below:

Platform	Option
LynxOS host and target	`-Dlynxos-gcc-x86`
Windows host LynxOS PowerPC target	`-Dlynxos-gcc-ppc-win32`
Linux host LynxOS PowerPC target	`-Dlynxos-gcc-ppc-linux`
Linux host LynxOS x86 target	`-Dlynxos-gcc-x86`

6.4.3 Linking

The Spectra ORB-required linking options are described in Section *5.2, Linking with Spectra ORB Libraries*, *on page 15*.

6.4.3.1 Creating Shared Libraries

The following LynxOS specific link options are needed when creating shared libraries. The options are in addition to any other link options or requirements.

```
-mshared -mthreads -shared
```

6.4.4 Deploying and Running

The procedures for resolving servers on LynxOS are the same as for Unix: see Section *6.1.4, Deploying and Running*, on page *17*, for details.

6.5 TI DSP/BIOS

The Spectrum Digital DSK 6455 board is the reference platform for this release, using the TI NDK TCP transport library and on board ethernet adapter. The Spectrum Digital 6416 boards can also be supported with either a DSign DSK-91C111 mezzanine ethernet adapter.

Only the C version of the ORB is supported on these platforms as it is more suited to the resource constrained DSP environment.

6.5.1 Environment

TI Code Composer Studio version 3.3 and NDK version 2.00 are supported for the DSK 6455 platform. Two `EORBENV` environment settings are supported.

- `dspbios-cl6x-win32` – fox 64xx series targets, Windows host.
- `dspbios-cl6x-linux` – for 64xx series targets, Linux host.

Code Composer Studio configuration files for the supported platforms are provided in the `etc/dspbios` directory. These files are used by the example make system to build examples.

Additional environment configuration variables are needed for building on both Linux and Windows host systems. These give the installation directories for BIOS, the tool chain (C6X), NDK, the chip support library (CSL) and the board support library (BSL):

Variable	Typical Value (Win32)	Typical Value (Linux)
`BIOS_INSTALL_DIR`	`C:/CCStudio_v3.3/bios_5_33_06`	`/opt/TI/bios_5_33_06`
`C6X_INSTALL_DIR`	`C:/CCStudio_v3.3/c6000/cgtools`	`/opt/TI/C6000CGT6.1.17`
`NDK_INSTALL_DIR`	`C:/CCStudio_v3.3/ndk_1_93_eval`	`/opt/TI/ndk_2_0_0`
`CSL_INSTALL_DIR`	`C:/CCStudio_v3.3/C6000`	`/opt/TI/C6xCSL`
`BSL_INSTALL_DIR`	`C:/CCStudio_v3.3/boards`	`/opt/TI/boards`

On Windows host systems the tool chain and BIOS versions are those used by default by Code Composer Studio v3.3; on Linux the latest released versions have been used. Please contact PrismTech for any specific version requirements.

Set `EORBHOME` to the directory where Spectra ORB is installed.

6.5.2 Installing

Extract the installation file to the directory of the host PC development system where you want Spectra ORB installed and for using with the Code Composer Studio IDE.

6.5.2.1 Software Requirements

The following software is required to run the ORB on the TI DSP/BIOS:

- Code Composer Studio v3.3 for c6000
- Network Developer Kit (NDK) for c6000

6.5.2.2 IDE Configuration

Example project files (`*.pjt`) require variable configuration and are defined in a distributed `macro.ini` file (located in `%EORBHOME%\etc\dspbios`). The `macro.ini` file needs to be modified to indicate where Code Composer Studio, NDK and Spectra ORB are installed. A copy of `macro.ini` must be placed in the directory where the Code Composer IDE binary (`cc_app.exe`) is executed from for the variable settings to be successfully read by Code Composer Studio.

6.5.3 Building

6.5.3.1 DSP Core Targets

The distribution has been tested on the TMS320C6416 DSK and C6455 DSK development boards. Please consult PrismTech for other DSP target builds.

6.5.4 Compiling

6.5.4.1 The IDL Compiler

The path to the `idlc` executable must be placed in host path environment variable in order for the host development environment to locate the IDL C compiler.

For Windows:

```
%EORBHOME%\bin\win32-msdev-x86
```

For Unix (or Unix variant):

```
$EORBHOME/bin/linux-gcc-x86
```

6.5.4.2 NDK TCP Library

The default NDK configuration supplied is built to use DHCP to configure the transport. To change this default behaviour or otherwise customise the transport the included `ndk.c` source file must be recompiled. To configure the NDK TCP library, edit the following file:

```
include/eOrbC/os/dspbios/ndk/ndk.h
```

Then recompile:

```
include/eOrbC/os/dspbios/ndk/ndk.c
```

replacing the generated object file in the `ec_tcp` transport library.

6.5.4.3 Dsign TCP Library

To configure the DSign TCP library, edit the following file:

```
include/eOrbC/os/dspbios/dsign/dsign.h
```

Then recompile:

```
include/eOrbC/os/dspbios/dsign/dsign.c
```

replacing the generated object file in the `ec_tcp` transport library.

6.5.5 Linking

The Spectra ORB-required linking options are described in Section *5.2, Linking with Spectra ORB Libraries, on page 15*. Use the linker provided with your platform's DSP tool when linking TI DSP/BIOS-based applications. Refer to your platform's documentation for linking instructions.

6.5.6 Deploying and Running

Deployment and running is specific to the DSP platform you are developing for and is usually managed by its IDE tool. Refer to your platform's documentation for deployment and running instructions.

6.5.6.1 Network Developer Kit (NDK)

In order to run examples successfully on NDK, a TCP/IP transport IP configuration is required for DSP. IP address settings for the DSK board can be found in

include/eOrbC/os/dspbios/ndk.h. The following settings must be modified for local IP address configuration:

```
#define EORB_HOSTNAME "c6416"
#define EORB_LOCALIPADDR "192.168.0.2
#define EORB_LOCALIPMASK "255.255.255.0
#define EORB_GATEWAYIP "192.168.0.1”
#define EORB_DOMAINNAME "prismtech.com”
#define EORB_DNSSERVER "0.0.0.0
```

6.6 Windows

Unzip the installation file to the directory where you want Spectra ORB to be installed.

6.6.1 Environment

Set the EORBHOME environment variable to the above directory and EORBENV to win32-msdev-x86.

Add the following to your path:

```
%EORBHOME%\bin\%EORBENV%
%EORBHOME%\lib\%EORBENV%
```

The win32-msdev-x86 configuration supports both Visual C++ versions 7 and 8. Version 6 is no longer supported.

6.6.2 Compiling and Linking

Applications using the win32-msdev-x86 distribution of Spectra ORB can be compiled and linked with *Microsoft Visual C++* using either the *MS Developer Studio* or from the command line with Microsoft's *cl.exe* utility.

! The *cl.exe* utility is unable to correctly interpret pathnames containing spaces which are given on the command line, even if they are wrapped in double quotes (“ ”). For example, cl.exe is unable to correctly interpret the directory path “C:\Program Files\MSDEV” since it contains a space between the words *Program* and *Files*.

When using Developer Studio (MSDEV), ensure Spectra ORB include and lib/<platform> directories are listed in the *Include files* and *Library* sections of the Directory tab of the Options dialogue. You can then compile and link using the standard MSDEV procedures.

6.6.2.1 Debug vs Release

Each ORB distribution comes with a set of debug libraries and a set of release libraries. The debug library names have a ***_g*** suffix in order to distinguish them from the standard release libraries. For example, the standard release version of the e_orb library is called e_orb.lib; the debug version is e_orb_g.lib.

Release

The ORB release libraries are compiled with the -MD compiler switch (note this is case sensitive). This ensures that they will be linked with Microsoft's multi-threaded, DLL, non-debug runtime library (mcvcrt.lib).

Debug

The ORB debug libraries are compiled with the -MDd compiler switch (note this is case sensitive). This ensures that they will be linked with Microsoft's multi-threaded, DLL, debug-enabled runtime

library (`msvcrtd.lib`).

All application code must be compiled with either `-MD` or `-MDd,` but not both, in order to ensure that the final executable file contains just one version of the C runtime library.

For example, if linking your application code with the ORB debug libraries (those with a `_g` suffix), you should compile your code using the `-MDd` switch.

i Do not add the `msvcrt.lib` or the `msvcrt.lib` libraries to the MSDEV library link list: linking to these libraries is performed automatically and transparently by the `/MD` and `/MDd` switches.

6.6.2.2 Creating a DLL

There are no ORB-specific requirements for switches for creating a DLL.

6.6.2.3 Creating an EXE

There are no ORB-specific requirements for switches for creating an EXE.

6.6.2.4 Exception Handling and Run-Time Type Information

The ORB is distributed with or without support for true C++ exceptions. Which type of distribution you have will depend on your requirements, but the version will also affect how your application code is compiled.

You must use the `-GX` *and* `-GR` compiler switches when compiling your application and linking a true C++ exception build of the ORB. This ensures that the compiler generates the appropriate stack unwinding and runtime type information support.

There are no specific compiler switches required when compiling application code which is linked with a non-exception build of the ORB. By default, the MSDEV compiler has exception handling and Run-Time Type Information (RTTI) disabled.

6.6.2.5 Compiler Options

The Spectra ORB-related compiler options should be set as described in Section *5*, *General Compilation Issues*, *on page 15*.

6.6.3 Deploying and Running

The procedures for deploying and running Spectra ORB-based applications on Windows platforms are the same as they are on Unix: follow the instructions given in Section 6.3.2, *Deploying and Running Applications* of the *C++ User Guide*, but replace the term *terminal window* with *command prompt*.

6.7 QNX

QNX can be treated as a UNIX system for native compilation. For cross compilation (typically from Linux), the host `lib` directory must be added to `LD_LIBRARY_PATH` and the host `bin` directory to `PATH`. This is to support the host based IDL compiler.

7 Licensing

7.1 General

For supported platforms this section describes how to install a license file for the ORB and how to use the license manager. The RLM product is used to manage licenses.

The licensing software is automatically installed on the host machine as part of the distribution. The software consists of two parts:

- Binary files installed in the following directory:

  `$EORBHOME/bin/$EORBENV`

- License file(s) which determine the terms of the license. These will be supplied by PrismTech.

Evaluation Licenses: PrismTech will supply an ORB evaluation license file, `license.lic`. This file is ***not*** included in the software distribution but is sent on request.

For node-locked licenses it is not necessary to run the license daemon, however for more complex licensing scenarios (such as floating licenses) it may be required.

7.1.1 Development and Deployment Licenses

Development licenses are on a *per Single Named Developer* basis. This implies that each developer using the product requires a license. The ORB is physically licensed for development purposes. Subsequently, each development license includes an IDL compiler license and an ORB library license.

7.2 Installing the License File

Copy the license file into the following location on the development host:

```
$EORBHOME/etc/license.lic
```

This is the recommended name and location for the license file but you can put the file or files in any accessible location. An environment variable, either `RLM_LICENSE` or `prismtech_LICENSE`, must be set to the full path and file name of the license file (either variable can be set; there is no need to set both). For example:

```
RLM_LICENSE=/my/lic/dir/license.lic
```

If licenses are distributed between multiple license files, the environment variable can be set to point to the directory which contains the license files. Alternatively multiple license files can be concatenated together into a single license file. If the default license location is used then no licensing environment variable need to be set.

7.3 Running the License Manager Daemon

It is only necessary to run the License Manager Daemon for floating or counted licenses. In this case, the license manager must be running before the ORB can be used. The license manager software is responsible for allocating licenses to developers and ensuring that the allowed number of concurrent licenses is not exceeded. To run the license manager, use the following command:

```
% rlm -c <location>
```

where <location> is either the full path name of the license file or the directory containing multiple license files. The `rlm` command will start the PrismTech vendor daemon `prismtech`, which controls the licensing of the ORB software.

To obtain a license for the ORB from a License Manager Daemon that is running on a different machine, set either the `RLM_LICENSE` or `prismtech_LICENSE` environment variable to point to the License Manager Daemon, using the following syntax:

```
% RLM_LICENSE=<port>@<host>
```

where `<port>` is the port the daemon is running on and `<host>` is the host the daemon is running on. The port and host values can be obtained from the information output when the daemon is started. The format of this output is as shown in the following example:

```
02/15 16:33 (rlm) License server started on azathoth
02/15 16:33 (rlm) Server architecture: x86_l2
02/15 16:33 (rlm) License files:
02/15 16:33 (rlm)     /home/steve/orbs/eorb16/etc/license.lic
02/15 16:33 (rlm)
02/15 16:33 (rlm) Web server starting on port 5054
02/15 16:33 (rlm) Using TCP/IP port 5053
02/15 16:33 (rlm) Starting ISV servers:
02/15 16:33 (rlm)     ... prismtech on port 59576
02/15 16:33 (prismtech) RLM License Server Version 9.3BL2 for ISV "prismtech"
02/15 16:33 (prismtech) Server architecture: x86_l2
...
02/15 16:33 (prismtech)
02/15 16:33 (prismtech) Server started on azathoth (hostid: 001ec95e393e) for:
02/15 16:33 (prismtech)         eorb_c eorb_c_idlc eorb_c_namsvc eorb_c_logsvc
02/15 16:33 (prismtech)         eorb_c_evtsvc eorb_cpp eorb_cpp_idlc
eorb_cpp_namsvc
02/15 16:33 (prismtech)         eorb_cpp_logsvc eorb_cpp_evtsvc
02/15 16:33 (prismtech)
02/15 16:33 (prismtech) License files:
02/15 16:33 (prismtech)     /home/steve/orbs/eorb16/etc/license.lic
02/15 16:33 (prismtech)
```

Here the value for `RLM_LICENSE` or `prismtech_LICENSE` would be *59576@azathoth*.

The license manager daemon also has a web management interface, simply point a browser at the server host on the port shown in the start up log (5054 in the example)

7.3.1 Utilities

A utility program, `rlmutil`, is available for license server management and administration. One feature of this utility is its ability to gracefully shut down the license manager. To shut down the license manager, preventing the checkout of licenses for the ORB software, run either of the following commands:

```
% rlmutil rlmdown prismtech
```

or

```
% rlmutil rlmdown -c <location>
```

where `<location>` is the full path and file name of the license file. The `rlmutil` program is also used to generate a host identification code which is used to generate your license key. To generate the code, run the following command on the license server:

```
% rlmutil rlmhostid
```

This returns an ID code for the server, which will look something like “001ec95e393e”. This ID code must be supplied to PrismTech so that your license key can be generated.

8 Documentation

The documentation for the C and C++ ORBs includes a number of documents which provide detailed information about the ORBs, their services, supported APIs, usage, installation and configuration.

The following table, lists all of the documentation and manuals included with the product. The table includes brief descriptions of the documents.

Document	File	Description
Product and Installation Guide	Install.pdf	This guide. Installation and configuration guide for the C and C++ ORBs.
Real-time User Guide	rt-guide.pdf	The real-time CORBA user guide for the C and C++ ORBs.
C IDL Guide	CIDLGuide.pdf	The C ORB IDL compiler guide which describes the IDL to C language mapping and how to use the IDL compiler (idlc)
C ORB User Guide	CUserGuide.pdf	The general user guide for the C ORB describing how to use the ORB and the features supported.
C ORB Reference Guide	CReferenceGuide.pdf	The reference guide for all the APIs and functions supported by the C ORB.
C ORB Lightweight Event Service Guide	CEventServiceLW.pdf	Describes the concepts, principles, and features of the C ORB Lightweight Event Service.
C ORB Lightweight Naming service Guide	CNamingServiceLW.pdf	Describes the concepts, principles, and features of the C ORB Lightweight Naming Service.
C ORB Lightweight Log Service Guide	CLogServiceLW.pdf	Describes the concepts, principles, and features of the C ORB Lightweight Log Service.
CPP IDL Guide	CppIDLGuide.pdf	The C++ ORB IDL compiler guide which describes the IDL to C++ language mapping and how to use the IDL compiler (idlcpp)
CPP User Guide	CppUserGuide.pdf	The general user guide for the C++ ORB describing how to use the ORB and the features supported.
CPP ORB Naming Service Guide	CppNamingService.pdf	Describes the concepts, principles, and features of the C++ ORB Naming Service.
CPP ORB Lightweight Naming Service Guide	CppNamingServiceLW.pdf	Describes the concepts, principles, and features of the C++ ORB Lightweight Naming Service.
CPP ORB Lightweight Event Service Guide	CppEventServiceLW.pdf	Describes the concepts, principles, and features of the C++ ORB Lightweight Event Service.
CPP ORB Lightweight Log Service Guide	CppLogServiceLW.pdf	Describes the concepts, principles, and features of the C++ ORB Lightweight Log Service.

9 Information Sources

9.1 Product Information

Links to useful technical information for PrismTech products, including Spectra ORB and associated components, are listed below. These links are provided for the reader's convenience and may become out-of-date if changes are made on the PrismTech Web site after publication of this guide. Nonetheless, these links should still be reachable from the main PrismTech Web page located at:

http://www.prismtech.com/

9.1.1 Knowledge Base

The PrismTech Knowledge Base is a collection of documents and resources intended to assist our customers in getting the most out of the Spectra products. The Knowledge Base has the most up-to-date information about bug fixes, product issues and technical support for difficulties that you may experience. The Knowledge Base can be found via the support link at the PrismTech main web page.

9.1.2 Additional Technical Information

Information provided by independent publishers, newsgroups, web sites, and organisations, such as the Object Management Group, can be found on the PrismTech Web site:

http://www.prismtech.com/

9.2 Support

PrismTech provides a range of product support, consultancy and educational programmes to help you from product evaluation and development, through to deployment of applications using Spectra ORB. The support programmes are designed to meet a customers' particular needs and range from a basic Standard programme to the Gold programme, which provides comprehensive, 24 x 7 support.

Detailed information about PrismTech product support services, general support contacts and enquiries are described on the PrismTech Support page reached via the PrismTech Home page at:

http://www.prismtech.com/

10 Build Environments

A wide variety of native and cross compiled hosts and targets are supported. For embedded systems please contact PrismTech for details of specific boards supported. Additional systems can be supported on request.

10.1 Supported native build environments

EORBENV	Description
`linux-gcc-x86`	Linux x86 system, `gcc` compiler.
`linux-gcc-ia64`	Linux Itanium system, `gcc` compiler.
`linux-gcc-ppc`	Linux PPC system, `gcc` compiler.
`linux-gcc-arm`	Linux ARM system, `gcc` compiler.
`win32-msdev-x86`	Windows x86 system, Visual Studio compiler.
`solaris-gcc-sparc`	Solaris SPARC system, `gcc` compiler.
`solaris-suncc-sparc`	Solaris SPARC system, Sun Workshop compiler.
`solaris-gcc-x86`	Solaris x86 system, `gcc` compiler.
`solaris-suncc-x86`	Solaris x86 system, Sun Workshop compiler.
`aix-xlc-power5`	AIX Power5 system, IBM `xlc` compiler.
`hpux-acc-ia64`	HPUX Itanium system, HP `acc` compiler.
`hpux-gcc-parisc`	HPUX PARISC system, `gcc` compiler.
`netbsd-gcc-x86`	NetBSD x86 system, `gcc` compiler.
`lynxos-gcc-x86`	LynxOS x86 system, `gcc` compiler.
`lynxos-gcc-ppc`	LynxOS PPC system, `gcc` compiler.
`qnx-qcc-x86`	QNX x86 system, `qcc` (`gcc`) compiler.

Note that each build environment may support various versions of the operating system and compiler. For exact details on supported versions and distributions please contact PrismTech.

10.2 Supported cross build environments

EORBENV	Description
linux-gcc-arm-linux	Linux x86 host, Linux ARM target, gcc compiler.
linux-gcc-ppc-linux	Linux x86 host, Linux PPC target, gcc compiler.
linux-gcc-x86-linux	Linux x86 host, Linux x86 target, gcc compiler.
lynxos-gcc-ppc-linux	Linux x86 host, LynxOS PowerPC target, gcc compiler
lynxos-gcc-ppc-win32	Windows x86 host, LynxOS PowerPC target, gcc compiler.
lynxos-gcc-x86-win32	Windows x86 host, LynxOS x86 target.
vxworks-64-gnu-PII-linux	Linux x86 host, VxWorks 6.4 Pentium II target.
vxworks-64-gnu-PIII-linux	Linux x86 host, VxWorks 6.4 Pentium III target.
vxworks-64-gnu-ppc604-linux	Linux x86 host, VxWorks 6.4 PPC604 target.
vxworks-64-gnu-ppc604-win32	Windows x86 host, VxWorks 6.4 PPC604 target.
vxworks-551-gnu-ppc604-win32	Windows x86 host, VxWorks 551 PPC604 target.
vxworks-551-gnu-ppc405-win32	Windows x86 host, VxWorks 551 PPC405 target.
vxworks-551-gnu-ppc860-win32	Windows x86 host, VxWorks 551 PPC860 target.
vxworks-551-gnu-ppc7410-win32	Windows x86 host, VxWorks 551 PPC7410 target.
vxworks-551-gnu-ppc8540-win32	Windows x86 host, VxWorks 551 PPC8540 target.
vxworks-551-gnu-PII-win32	Windows x86 host, VxWorks 551 Pentium II target.
vxworks-551-gnu-PIII-win32	Windows x86 host, VxWorks 551 Pentium III target.
vxworks-551-gnu-sh-win32	Windows x86 host, VxWorks 551 SuperH target.
vxworks-551-gnu-ppc603-solaris	Solaris host, VxWorks PPC603 target.
vxworks-551-gnu-ppc604-solaris	Solaris host, VxWorks PPC604 target.
vxworks-67-gnu-arm-linux	Linux host, ARM target, gcc compiler.
vxworks-67-gnu-x86-linux	Linux host, x86 target, gcc compiler.
vxworks-67-gnu-ppc-linux	Linux host, PPC target, gcc compiler.
vxworks-67-gnu-arm-win32	Windows host, ARM target, gcc compiler.
vxworks-67-gnu-x86-win32	Windows host, x86 target, gcc compiler.
vxworks-67-gnu-ppc-win32	Windows host, PPC target, gcc compiler.
vxworks-68-gnu-arm-linux	Linux host, ARM target, gcc compiler.
vxworks-68-gnu-x86-linux	Linux host, x86 target, gcc compiler.
vxworks-68-gnu-ppc-linux	Linux host, PPC target, gcc compiler.
vxworks-68-gnu-arm-win32	Windows host, ARM target, gcc compiler.
vxworks-68-gnu-x86-win32	Windows host, x86 target, gcc compiler.
vxworks-68-gnu-ppc-win32	Windows host, PPC target, gcc compiler.
vxworks-69-gnu-arm-linux	Linux host, ARM target, gcc compiler.
vxworks-69-gnu-x86-linux	Linux host, x86 target, gcc compiler.

EORBENV	Description
vxworks-69-gnu-ppc-linux	Linux host, PPC target, gcc compiler.
vxworks-69-gnu-arm-win32	Windows host, ARM target, gcc compiler.
vxworks-69-gnu-x86-win32	Windows host, x86 target, gcc compiler.
vxworks-69-gnu-ppc-win32	Windows host, PPC target, gcc compiler.
integrity-multi405-x86	Linux x86 host, Integrity 405 x86 target.
integrity-multi405-ppc	Linux x86 host, Integrity 405 PPC target.
integrity-multi405-arm	Linux x86 host, Integrity 405 ARM target.
integrity-multi407-x86	Linux x86 host, Integrity 505 x86 target.
integrity-multi407-ppc	Linux x86 host, Integrity 505 PPC target.
integrity-multi421-x86	Linux x86 host, Integrity 506 x86 target.
integrity-multi421-ppc	Linux x86 host, Integrity 506 PPC target.
integrity-multi423-ppc	Linux x86 host, Integrity 507 PPC target.
integrity-multi423-arm	Linux x86 host, Integrity 507 ARM target.
integrity-multi423-ppc	Linux x86 host, Integrity 508 PPC target.
dspbios-cl6x-win32	Windows x86 host, c64xx series target.
dspbios-cl6x-linux	Linux x86 host, c64xx series target.
qnx-qcc-linux	Linux x86 host, QNX target.