RTI Connext DDS

Core Libraries

Platform Notes

Version 5.2.3

Printed in U.S.A. First printing.
April 2016.

Trademarks

Copy and Use Restrictions

Technical Support

Real-Time Innovations, Inc.
232 E. Java Drive
Sunnyvale, CA 94089
Phone: (408) 990-7444
Email: support@rti.com
Website: https://support.rti.com/

Chapter 1 Supported Platforms

1.1 Paths Mentioned in Documentation 4

Chapter 2 AIX Platforms

2.1 Support for Modern C++ API 11

2.2 Multicast Support 11

2.3 Transports 12

2.3.1 Notes for Using Shared Memory 12

2.4 Monotonic Clock Support 12

2.5 Thread Configuration 12

2.5.1 Changing Thread Priority 13

2.5.2 Support for Controlling CPU Core Affinity for RTI Threads 13

2.6 Durable Writer History and Durable Reader State Features 14

2.7 Distributed Logger Support 15

2.8 Libraries Required for Using Monitoring 15

Chapter 3 Android Platforms

3.1 Support for Modern C++ API 21

3.2 Multicast Support 21

3.3 Transports 21

3.4 Monotonic Clock Support 22

3.5 Thread Configuration 22

3.5.1 Support for Controlling CPU Core Affinity for RTI Threads 23

3.6 Durable Writer History and Durable Reader State Features 24

3.7 Distributed Logger Support 24

3.8 Libraries Required for Using Monitoring 24

3.9 Libraries Required for Using RTI Secure WAN Transport APIs 24

3.10 Libraries Required for Using RTI TCP Transport APIs 25

Chapter 4 INTEGRITY Platforms

4.1 Required Patches for INTEGRITY 10.0.2 and 11.0.4 30

4.2 Support for Modern C++ API 30

4.3 Multicast Support 30

4.4 Supported Transports 30

4.5 Monotonic Clock Support 30

4.6 Thread Configuration 31

4.6.1 Socket-Enabled and POSIX-Enabled Threads are Required 33

4.6.2 Support for Controlling CPU Core Affinity for RTI Threads 33

4.7 Durable Writer History and Durable Reader State Features 33

4.8 Libraries Required for Using Distributed Logger 33
4.9 Libraries Required for Using Monitoring 34
4.10 Request-Reply Communication Pattern 34
4.11 Diagnostics on INTEGRITY Systems 35
4.12 Running over IP Backplane on a Dy4 Champ-AVII Board 35
4.13 Multi-NIC Support on INTEGRITY 5.0 35
4.14 Out-of-the-box Transport Compatibility with Other Connext DDS Platforms 35
4.14.1 Smaller Shared-Memory Receive-Resource Queue Size 36
4.14.2 Using Shared Memory on INTEGRITY Systems 36
4.14.3 Shared Memory Limitations on INTEGRITY Systems 37
4.15 Using rtiddsping and rtiddsspy on PowerPC INTEGRITY Systems 37
4.16 Issues with INTEGRITY Systems 38
4.16.1 Delay When Writing to Unreachable Peers 38
4.16.2 Linking with 'libivfs.a' without a File System 38
4.16.3 Compiler Warnings Regarding Unrecognized #pragma Directives 38
4.16.4 Warning when Loading Connext DDS Applications on INTEGRITY Systems 38

Chapter 5 iOS Platforms

5.1 Supported Languages 42
5.1.1 Support for Modern C++ API 42
5.2 Multicast Support 42
5.3 Transports 42
5.4 Unsupported Features 43
5.5 Thread Configuration 43
5.5.1 Support for Controlling CPU Core Affinity for RTI Threads 43
5.6 Libraries Required for Using Distributed Logger 45
5.7 Libraries Required for Using Monitoring 45
5.8 Libraries Required for Using RTI Secure WAN Transport 46

Chapter 6 Linux Platforms

6.1 Support for Modern C++ API 58
6.2 Multicast Support 58
6.3 Supported Transports 58
6.3.1 Shared Memory Support 59
6.4 Monotonic Clock Support 59
6.5 Thread Configuration 59
6.5.1 Native POSIX Thread Library (NPTL) Requirements 59
6.5.2 Support for Controlling CPU Core Affinity for RTI Threads 60

6.6 Durable Writer History and Durable Reader State Features 62
6.7 Libraries Required for Using Distributed Logger 63
6.8 Libraries Required for Using Monitoring 64
6.9 Libraries Required for Using RTI Secure WAN Transport APIs 64
6.10 Libraries Required for Using RTI TCP Transport APIs 65
Chapter 7 LynxOS Platforms
7.1 Support for Modern C++ API 72
7.2 Multicast Support 72
7.3 Supported Transports 73
7.3.1 Shared Memory Support 73
7.4 Monotonic Clock Support 73
7.5 Thread Configuration 73
7.5.1 Support for Controlling CPU Core Affinity for RTI Threads 74
7.6 Durable Writer History and Durable Reader State Features 75
7.7 Distributed Logger Support 75
7.8 Libraries Required for Using Monitoring 75
7.9 IP Fragmentation Issues 76
Chapter 8 OS X Platforms
8.1 Support for Modern C++ API 82
8.2 Multicast Support 83
8.3 Supported Transports 83
8.4 Monotonic Clock Support 83
8.5 Thread Configuration 83
8.5.1 Support for Controlling CPU Core Affinity for RTI Threads 83
8.6 Durable Writer History and Durable Reader State Features 85
8.7 Libraries Required for Using Distributed Logger 85
8.8 Libraries Required for Using Monitoring 85
8.9 Libraries Required for Using RTI Secure WAN Transport APIs 86
8.10 Libraries Required for Using RTI TCP Transport APIs 87
Chapter 9 QNX Platforms
9.1 Required Change for Building with C++ Libraries for QNX Platforms 91
9.2 Support for Modern C++ API 92
9.3 Multicast Support 92
9.4 Supported Transports 92
9.5 Monotonic Clock Support 93
9.6 Thread Configuration 93

9.6.1 Support for Controlling CPU Core Affinity for RTI Threads 93
9.7 Durable Writer History and Durable Reader State Features 94
9.8 Libraries Required for Using Distributed Logger 95
9.9 Libraries Required for Using Monitoring 95
9.10 Libraries Required for Using RTI Secure WAN Transport APIs 96
9.11 Libraries Required for Using RTI TCP Transport APIs 96
9.12 Restarting Applications on QNX Systems 97
Chapter 10 Solaris Platforms
10.1 Request-Reply Communication Pattern 102
10.2 Support for Modern C++ API 103
10.3 Multicast Support 103
10.4 Supported Transports 103
10.4.1 Shared Memory Support 103
10.4.2 Increasing Available Shared Resources 104
10.5 Monotonic Clock Support 104
10.6 Thread Configuration 105
10.6.1 Support for Controlling CPU Core Affinity for RTI Threads 105
10.7 Durable Writer History and Durable Reader State Features 106
10.8 Distributed Logger Support 106
10.9 Libraries Required for Using Monitoring 107
10.10 Libraries Required for using RTI Secure WAN Transport APIs 107
Chapter 11 VxWorks Platforms
11.1 Required Makefile Change for VxWorks 7.0 Platforms 125
11.2 Request-Reply Communication Pattern 125
11.3 Increasing the Stack Size 126
11.4 Libraries for RTP Mode on VxWorks 6.3 and Higher Systems 126
11.5 Requirement for Restarting Applications 126
11.6 Support for Modern C++ API 127
11.7 Multicast Support 127
11.8 Supported Transports 127
11.8.1 Shared-Memory Communication between Applications Running in Kernel Mode and RTP Requires Explicitly Set Participant ID 128
11.8.2 How To Run Connext DDS Libraries in Kernels Built without Shared Memory 128
11.9 Monotonic Clock Support 129
11.10 Thread Configuration 129
11.10.1 Support for Controlling CPU Core Affinity for RTI Threads 129
11.11 Durable Writer History and Durable Reader State Features 131

11.12 Libraries Required for Using Distributed Logger 131
11.13 Libraries Required for Using Monitoring 132
11.14 Increasing the Receive Socket Buffer Size 132
Chapter 12 Windows Platforms
12.1 Requirements when Using Microsoft Visual Studio 152
12.2 Windows Registry Setting for Better Performance 153
12.3 Use Dynamic MFC Library, Not Static 154
12.4 .NET API Requires Thread Affinity 154
12.5 ODBC Database Compatibility 154
12.6 PPP Link Support for Windows XP Systems 155
12.7 Support for Modern C++ API 155
12.8 Multicast Support 156
12.9 Supported Transports 156
12.10 Monotonic Clock Support 156
12.11 Thread Configuration 156
12.11.1 Support for Controlling CPU Core Affinity for RTI Threads 158
12.12 Durable Writer History and Durable Reader State Features 158
12.13 Libraries Required for Using Distributed Logger Support 159
12.14 Libraries Required for Using Monitoring 159
12.15 Libraries Required for Using RTI Secure WAN Transport APIs 159
12.16 Libraries Required for Using RTI TCP Transport APIs 160

Chapter 1 Supported Platforms

This document provides platform-specific instructions on how to compile, link, and run *RTI® Connext™ DDS* applications.

Connext DDS 5.2.3 is available on all the platforms in Table 1.1 Available Platforms If you want to use a platform that is not on RTI's download portal, please contact RTI Support.

Table 1.1 Available Platforms

Operating System		Reference
AIX®	AIX 5.3, 7.1	AIX Platforms (Chapter 2 on page 6)
Android™	Android 2.3 - 4.4, 5.0, 5.1	Android Platforms (Chapter 3 on page 16)
INTEGRITY®	INTEGRITY 5.0.11, 10.0.2, and 11.0.4	INTEGRITY Platforms (Chapter 4 on page 26)
iOS	iOS 8.2	iOS Platforms (Section Chapter 5 on page 40)

Table 1.1 Available Platforms

Operating System		Reference
Linux® (ARM® CPU)	Raspbian Wheezy 7.0 (3.x kernel)	Linux Platforms (Chapter 6 on page 47)
Linux (Intel® CPU)	CentOS 5.4, 5.5, 6.0, 6.2 - 6.4, 7.0 Red Hat® Enterprise Linux 4.0, 5.0-5.2, 5.4, 5.5, 6.0 - 6.5, 6.7, 7.0 SUSE® Linux Enterprise Server 11 SP2, SP3 Ubuntu® Server 12.04 LTS, Ubuntu 14.04 LTS Wind River® Linux 4	
Linux (PowerPC® CPU)	Freescale P2020RDB Wind River Linux 3 Yellow Dog™ Linux 4.0	
LynxOS®[a]	LynxOS 4.0, 4.2, 5.0	LynxOS Platforms (Chapter 7 on page 67)
Mac® OS X	OS X 10.8, 10.10, 10.11	OS X Platforms (Chapter 8 on page 77)
QNX®	QNX Neutrino® 6.4.1, 6.5	QNX Platforms (Chapter 9 on page 88)
Solaris™	Solaris 2.9, 2.10	Solaris Platforms (Chapter 10 on page 98)
VxWorks®	VxWorks 5.5, 6.3 - 6.9, 6.9.3.2, 6.9.4, 7.0 VxWorks 653 2.3	VxWorks Platforms (Chapter 11 on page 109)
Windows®	Windows 7, 8, 8.1, 10 Windows Server 2003, 2008 R2, 2012 R2 Windows Vista® Windows XP Professional SP2	Windows Platforms (Chapter 12 on page 138)

For each platform, this document provides information on:

- Supported operating systems and compilers
- Required Connext DDS and system libraries
- Required compiler and linker flags

[a]The Java API is not supported on LynxOS platforms. If your application requires support for Java on LynxOS, please contact your RTI account manager.

- Required environment variables for running the application (if any)
- Details on how the Connext DDS libraries were built
- Support for the Modern C++ API
- Multicast support
- Supported transports
- Monotonic clock support
- Thread configuration
- Durable Writer History and Durable Reader State features support

1.1 Paths Mentioned in Documentation

The documentation refers to:

- **<NDDSHOME>**

  This refers to the installation directory for Connext DDS. The default installation paths are:
  - Mac OS X systems:
    /Applications/rti_connext_dds-5.2.3
  - UNIX-based systems, non-*root* user:
    /home/your user name/rti_connext_dds-5.2.3
  - UNIX-based systems, *root* user:
    /opt/rti_connext_dds-5.2.3
  - Windows systems, user without Administrator privileges:
    <your home directory>\rti_connext_dds-**5.2.3**
  - Windows systems, user with Administrator privileges:
    C:\Program Files\rti_connext_dds-5.2.3 (64-bit machines)
    C:\Program Files (x86)\rti_connext_dds-5.2.3 (32-bit machines)

  You may also see $NDDSHOME or %NDDSHOME%, which refers to an environment variable set to the installation path.

  Wherever you see <NDDSHOME> used in a path, replace it with your installation path.

  Note for Windows Users: When using a command prompt to enter a command that includes the path **C:\Program Files** (or any directory name that has a space), enclose the path in quotation marks. For example:

  ```
  "C:\Program Files\rti_connext_dds-5.2.3\bin\rtiddsgen"
  ```

  Or if you have defined the NDDSHOME environment variable:

  ```
  "%NDDSHOME%\bin\rtiddsgen"
  ```

- **<path to examples>**

By default, examples are copied into your home directory the first time you run *RTI Launcher* or any script in **<NDDSHOME>/bin**. This document refers to the location of the copied examples as <path to examples>.

Wherever you see <path to examples>, replace it with the appropriate path.

Default path to the examples:

- Mac OS X systems: **/Users/*your user name*/rti_workspace/5.2.3/examples**
- UNIX-based systems: **/home/*your user name*/rti_workspace/5.2.3/examples**
- Windows systems: ***your Windows documents folder*\rti_workspace\5.2.3\examples**

  Where 'your Windows documents folder' depends on your version of Windows. For example, on Windows 7, the folder is **C:\Users\your user name\Documents**; on Windows Server 2003, the folder is **C:\Documents and Settings\your user name\Documents**.

Note: You can specify a different location for **rti_workspace**. You can also specify that you do not want the examples copied to the workspace. For details, see *Controlling Location for RTI Workspace and Copying of Examples* in the *Connext DDS Core Libraries Getting Started Guide*.

Chapter 2 AIX Platforms

Table 2.1 Supported AIX Target Platforms lists the architectures supported on the IBM® AIX operating system.

Table 2.1 Supported AIX Target Platforms

Operating System	CPU	Compiler	RTI Architecture Abbreviation
AIX 5.3	POWER5 (32-bit mode)	IBM XLC for AIX v9.0	p5AIX5.3xlc9.0
	POWER5 (64-bit mode)	IBM XLC for AIX v9.0	64p5AIX5.3xlc9.0
AIX 7.1	POWER7 (32-bit mode)	IBM xlC_r for AIX v12.1	p7AIX7.1xlc12.1
		IBM Java 1.7 or 1.8	
	POWER7 (64-bit mode)	IBM xlC_r for AIX v12.1	64p7AIX7.1xlc12.1
		IBM Java 1.7 or 1.8	

Table 2.2 Building Instructions for AIX Architectures lists the compiler flags and the libraries you will need to link into your application. See also: Libraries Required for Using Monitoring (Section 2.8 on page 15)

Make sure you are consistent in your use of static, dynamic, debug and release versions of the libraries. Do not link both static and dynamic libraries. Similarly, do not mix release and debug libraries.

Table 2.3 Running Instructions for AIX Architectures provides details on the environment variables that must be set at run time for an AIX architecture.

Table 2.4 Library-Creation Details for AIX Architectures provides details on how the libraries were built. This table is provided strictly for informational purposes; you do not need to use these

parameters to compile your application. You may find this information useful if you are involved in any in-depth debugging.

Table 2.2 Building Instructions for AIX Architectures

API	Library Format	Required RTI Libraries[ab] [c]	Required System Libraries[d]	Required Compiler Flags
C++ (Traditional and Modern APIs)	Static Release	libnddscppz.a or libnddscpp2z.a libnddscz.a libnddscorez.a librticonnextmsgcppz.a	-ldl -lnsl -lm -pthread	-DRTI_AIX -DRTI_UNIX -q[32|64][e] -qlongdouble
	Static Debug	libnddscppzd.a or libnddscpp2zd.a libnddsczd.a libnddscorezd.a librticonnextmsgcppzd.a		
	Dynamic Release	libnddscpp.so or libnddscpp2.so libnddsc.so libnddscore.so librticonnextmsgcpp.so	-ldl -lnsl -lm -pthread -brtl	
	Dynamic Debug	libnddscppd.so or libnddscpp2d.so libnddscd.so libnddscored.so librticonnextmsgcppd.so		

[a]Choose libnddscpp*.* for the Traditional C++ API or libnddscpp2*.* for the Modern C++ API.

[b]Connext DDS C/C++ libraries are in ${NDDSHOME}/lib/<***architecture***>. NDDSHOME is where Connext DDS is installed, see **Paths Mentioned in Documentation (Section 1.1 on page 4)**

[c]The ***rticonnextmsg*** library only applies if you have the RTI Connext DDS Professional, Evaluation, or Basic package type. It is not provided with the RTI Connext DDS Core package type.

[d]Transports (other than the default IP transport) such as StarFabric may require linking in additional libraries. For details, see the API Reference HTML documentation or contact support@rti.com.

[e]Use '-q32' if you build 32-bit code or '-q64' for 64-bit code.

Table 2.2 Building Instructions for AIX Architectures

API	Library Format	Required RTI Libraries[ab] [c]	Required System Libraries[d]	Required Compiler Flags
C	Static Release	libnddscz.a libnddscorez.a librticonnextmsgcz.a	-ldl -lnsl -lm -pthread	-DRTI_AIX -DRTI_UNIX -q[32|64][e] -qlongdouble -qthreaded [f]
	Static Debug	libnddsczd.a libnddscorezd.a librticonnextmsgczd.a		
	Dynamic Release	libnddsc.so libnddscore.so librticonnextmsgc.so	-ldl -lnsl -lm -pthread -brtl	
	Dynamic Debug	libnddscd.so libnddscored.so librticonnextmsgcd.so		
Java	Release	nddsjava.jar rticonnextmsg.jar	N/A	N/A
	Debug	nddsjavad.jar rticonnextmsgd.jar		

[a] Choose libnddscpp*.* for the Traditional C++ API or libnddscpp2*.* for the Modern C++ API.

[b] Connext DDS C/C++ libraries are in ${NDDSHOME}/lib/<***architecture***>. NDDSHOME is where Connext DDS is installed, see **Paths Mentioned in Documentation (Section 1.1 on page 4)**

[c] The ***rticonnextmsg*** library only applies if you have the RTI Connext DDS Professional, Evaluation, or Basic package type. It is not provided with the RTI Connext DDS Core package type.

[d] Transports (other than the default IP transport) such as StarFabric may require linking in additional libraries. For details, see the API Reference HTML documentation or contact support@rti.com.

[e] Use '-q32' if you build 32-bit code or '-q64' for 64-bit code.

[f] The '-qthreaded' option is automatically set if you use one of the compilers that ends with '_r', such as cc_r, xlc_r, xlC_r. See the IBM XLC reference manual for more information.

Table 2.3 Running Instructions for AIX Architectures

RTI Architecture	Library Format (Release & Debug)	Required Environment Variables[ab]
All supported AIX architectures for Java	N/A	LIBPATH=$(NDDSHOME)/lib/<*arch*>: $(LIBPATH) EXTSHM=ON
All other supported architectures	Static	EXTSHM=ON
	Dynamic	LIBPATH=$(NDDSHOME)/lib/<*arch*>: $(LIBPATH) EXTSHM=ON

[a] See **Notes for Using Shared Memory (Section 2.3.1 on page 12)**

[b] ${NDDSHOME} represents the root directory of your Connext DDS installation. ${LIBPATH} represents the value of the LIBPATH variable prior to changing it to support Connext DDS. When using nddsjava.jar, the Java virtual machine (JVM) will attempt to load release versions of the native libraries (nddsjava.so, nddscore.so, nddsc.so). When using nddsjavad.jar, the JVM will attempt to load debug versions of the native libraries (nddsjavad.so, nddscored.so, nddscd.so).

Table 2.4 Library-Creation Details for AIX Architectures

RTI Architecture	Library Format (Static & Dynamic)	Compiler Flags Used by RTI [a]	Libraries Used by RTI [b]
p5AIX5.3xlc9.0	Release	-q32 -qlongdouble -qalias=noansi -qpic=large -qthreaded -D_POSIX_C_ SOURCE=199506L -D__EXTENSIONS__ -O -qflag=i:i -DPtrIntType=long -DCSREAL IS_FLOAT -DCPU=Power5+ -DNDEBUG	
	Debug	-q32 -qlongdouble -qalias=noansi -qpic=large -qthreaded -D_POSIX_C_ SOURCE=199506L -D__EXTENSIONS__ -O -qflag=i:i -DPtrIntType=long -DCSREAL_IS_FLOAT -DCPU=Power5+ -g	
p7AIX7.1xlc12.1	Release	-q32 -qwarn64 -qlongdouble -qalias=noansi -qpic=large -qthreaded -D_POSIX_C SOURCE=199506L -D EXTENSIONS -O -qflag=i:i -DPtrIntType=long -DCSREAL_ IS_FLOAT -DCPU=Power7+ -DNDEBUG	-lC128
	Debug	-q32 -qwarn64 qlongdouble -qalias=noansi -qpic=large -qthreaded -D_POSIX_C_ SOURCE=199506L -D EXTENSIONS -O -qflag=i:i -DPtrIntType=long -DCSREAL_ IS_FLOAT -DCPU=Power7+ -g	-lC128
64p5AIX5.3xlc9.0	Release	-q64 -qwarn64 -qlongdouble -qalias=noansi -qpic=large -qthreaded -D_POSIX_C_ SOURCE=199506L -D__EXTENSIONS__ -O -qflag=i:i -DPtrIntType=long -DCSREAL_IS_FLOAT -DCPU=Power5+ -DNDEBUG	
	Debug	-q64 -qwarn64 -qlongdouble -qalias=noansi -qpic=large -qthreaded -D_POSIX_C_ SOURCE=199506L -D__EXTENSIONS__ -O -qflag=i:i -DPtrIntType=long -DCSREAL_IS_FLOAT -DCPU=Power5+ -g	

[a] Connext DDS was built using the 'xlC_r' compiler. See IBM's XLC reference manual for a description of the different compilers. For a list of the additional settings (defined by default) for the 'xlC_r' compiler, see the file **/etc/vac.cfg.53**.

[b] Linking without the 128-bit versions of the C Runtime Library when your program uses 128-bit long doubles (for example, if you specify -qldbl128 or -qlongdouble alone) may produce unpredictable results. Therefore, RTI libraries compiled with -qlongdouble are linked using -lC128. For more information, please consult the IBM compiler reference website: **http://pic.dhe.ibm.com/infocenter/comphelp/v121v141/index.jsp?topic=%2Fcom.ibm.xlcpp121.aix.doc%2Fcompiler_ref%2Fopt_ldbl128.html**

Table 2.4 Library-Creation Details for AIX Architectures

RTI Architecture	Library Format (Static & Dynamic)	Compiler Flags Used by RTI [a]	Libraries Used by RTI [b]
64p7AIX7.1xlc12.1	Release	-q64 -qwarn64 -qlongdouble -qalias=noansi -qpic=large -qthreaded -D_POSIX_C_SOURCE=199506L -D EXTENSIONS -O -qflag=i:i -DPtrIntType=long -DCSREAL_IS_FLOAT -DCPU=Power7+ -DNDEBUG	-lC128
	Debug	-q64 -qwarn64 qlongdouble -qalias=noansi -qpic=large -qthreaded -D_POSIX_C_SOURCE=199506L -D EXTENSIONS -O -qflag=i:i -DPtrIntType=long -DCSREAL_IS_FLOAT -DCPU=Power7+ -g	-lC128
All supported AIX architectures for Java	Release	-target 1.4 -source 1.4	
	Debug	-target 1.4 -source 1.4 -g	

2.1 Support for Modern C++ API

Connext DDS provides two different C++ APIs, which we refer to as the "Traditional C++" and "Modern C++" APIs. The Modern C++ API is available for all AIX 7.1 platforms.

For more information on Modern C++, see "Traditional vs. Modern C++" in the User's Manual.

2.2 Multicast Support

Multicast is supported on all AIX platforms and is configured out of the box. That is, the default value for the initial peers list (NDDS_DISCOVERY_PEERS) includes a multicast address. See the API Reference HTML documentation for more information.

---

[a] Connext DDS was built using the 'xlC_r' compiler. See IBM's XLC reference manual for a description of the different compilers. For a list of the additional settings (defined by default) for the 'xlC_r' compiler, see the file **/etc/vac.cfg.53**.

[b] Linking without the 128-bit versions of the C Runtime Library when your program uses 128-bit long doubles (for example, if you specify -qldbl128 or -qlongdouble alone) may produce unpredictable results. Therefore, RTI libraries compiled with -qlongdouble are linked using -lC128. For more information, please consult the IBM compiler reference website: **http://pic.dhe.ibm.com/infocenter/comphelp/v121v141/index.jsp?topic=%2Fcom.ibm.xlcpp121.aix.doc%2Fcompiler_ref%2Fopt_ldbl128.html**

2.3 Transports

- **Shared memory:** Supported and enabled by default.
- **UDPv4:** Supported and enabled by default.
- **UDPv6:** Not supported.
- **TCP/IPv4:** Not supported.

2.3.1 Notes for Using Shared Memory

By default, the maximum number of shared memory segments you can use with AIX is quite small and limits the capability of Connext DDS applications to work properly over shared memory. To increase the maximum number of shared memory segments an application can use, set the following environment variable before invoking your Connext DDS application:

```
EXTSHM=ON
```

This environment variable is not required if your application does not use the shared memory transport.

To see a list of shared memory resources in use, please use the '**ipcs**' command. To clean up shared memory and shared semaphore resources, please use the '**ipcrm**' command.

The shared memory keys used by Connext DDS are in the range of 0x400000. For example:

```
ipcs -m | grep 0x004
```

The shared semaphore keys used by Connext DDS are in the range of 0x800000; the shared mutex keys are in the range of 0xb00000. For example:

```
ipcs -s | grep 0x008
ipcs -s | grep 0x00b
```

Please refer to the shared-memory transport online documentation for details on the shared memory and semaphore keys used by Connext DDS.

2.4 Monotonic Clock Support

The monotonic clock (described in "Clock Selection" in the User's Manual) is not supported on AIX architectures.

2.5 Thread Configuration

Table 2.5 Thread Settings for AIX Platforms lists the thread settings for AIX platforms.

Table 2.6 Thread-Priority Definitions for AIX Platforms lists the thread-priority definitions for AIX platforms.

2.5.1 Changing Thread Priority

Due to the AIX threading-model implementation, there are situations that require you to run your Connext DDS application with root privileges:

- **For all APIs:** Your application must have *root* privileges to use the thread option, DDS_THREAD_SETTINGS_REALTIME_PRIORITY, for the event and receiver pool thread QoS (**DDS_DomainParticipantQos.event.thread, DDS_DomainParticipantQos.receiver_pool.thread**).
- **For the Java API only:** Your application must have *root* privileges to change the event and receiver pool thread priorities **(DDS_DomainParticipantQos.event.thread, DDS_DomainParticipantQos.receiver_pool.thread**).

2.5.2 Support for Controlling CPU Core Affinity for RTI Threads

Support for controlling CPU core affinity (described in "Controlling CPU Core Affinity" in the User's Manual) is not available for AIX platforms.

Table 2.5 Thread Settings for AIX Platforms

Applicable Thread	DDS_ThreadSettings_t	Platform-Specific Setting
Asynchronous Publisher, Asynchronous flushing thread	mask	OS default thread type
	priority	OS default thread priority
	stack_size	192*1024
	cpu_list	CPU core affinity not supported
	cpu_rotation	CPU core affinity not supported
Database thread	mask	DDS_THREAD_SETTINGS_STDIO
	priority	OS default thread priority
	stack_size	192*1024
	cpu_list	CPU core affinity not supported
	cpu_rotation	CPU core affinity not supported

Table 2.5 Thread Settings for AIX Platforms

Applicable Thread	DDS_ ThreadSettings_ t	Platform-Specific Setting
Event thread	mask	DDS_THREAD_SETTINGS_STDIO | DDS_THREAD_ SETTINGS_FLOATING_POINT
	priority	OS default thread priority
	stack_size	4*192*1024
	cpu_list	CPU core affinity not supported
	cpu_rotation	CPU core affinity not supported
ReceiverPool threads	mask	DDS_THREAD_SETTINGS_STDIO | DDS_THREAD_ SETTINGS_FLOATING_POINT
	priority	OS default thread priority
	stack_size	4*192*1024
	cpu_list	CPU core affinity not supported
	cpu_rotation	CPU core affinity not supported

Table 2.6 Thread-Priority Definitions for AIX Platforms

Thread-Priority Definition	Operating-System Priority
THREAD_PRIORITY_DEFAULT	If any of these constants are used to set the priority of the thread in the QoS, the OS's default thread priority will be used.
THREAD_PRIORITY_HIGH	
THREAD_PRIORITY_ABOVE_NORMAL	
THREAD_PRIORITY_NORMAL	
THREAD_PRIORITY_BELOW_NORMAL	
THREAD_PRIORITY_LOW	

2.6 Durable Writer History and Durable Reader State Features

The Durable Writer History and Durable Reader State features are not supported on AIX platforms.

2.7 Distributed Logger Support

RTI Distributed Logger is not supported on AIX platforms.

2.8 Libraries Required for Using Monitoring

Make sure you are consistent in your use of static, dynamic, debug and release versions of the libraries. For example, if your Connext DDS application is linked with the static release version of the Connext DDS libraries, you will need to also use the static release version of the monitoring library. Do not link both static and dynamic libraries. Similarly, do not mix release and debug libraries.

If you are statically linking your application with DDS libraries and you want to add monitoring to your application, you will also need to statically link the monitoring library. The library cannot be loaded dynamically strictly through the QoS profile because it also depends on DDS to publish its data. Therefore, it depends on DDS; the DDS functionality would cause duplicate symbols to be found resulting, in the termination of the process.

Note: If you plan to use *static* libraries, the RTI library from Table 2.7 Additional Libraries for Using Monitoring must appear *first* in the list of libraries to be linked.

Table 2.7 Additional Libraries for Using Monitoring

Static Release	Static Debug	Dynamic Release	Dynamic Debug
librtimonitoringz.a	librtimonitoringzd.a	librtimonitoring.so	librtimonitoringd.so

Chapter 3 Android Platforms

Table 3.1 Supported Android Target Platforms lists the architectures supported on the Android® operating system.

Table 3.1 Supported Android Target Platforms

Operating System	CPU	Compiler	RTI Architecture Abbreviation
Android 2.3 - 4.4	ARMv7a	gcc 4.8 (NDK r9)[a]	armv7aAndroid2.3gcc4.8
		Java Platform, Standard Edition JDK 1.7 or 1.8[b]	
Android 5.0, 5.1	ARMv7A	gcc 4.9 (NDK r10e)[c]	armv7aAndroid5.0gcc4.9ndkr10e
		Java Platform, Standard Edition JDK 1.7 or 1.8[d]	

SeeTable 3.2 Building Instructions for Android Architectures for a list of the compiler flags and libraries you will need to link into your application.

See also:

- Libraries Required for Using RTI TCP Transport APIs (Section 3.10 on page 25)
- Libraries Required for Using Monitoring (Section 3.8 on page 24)

---

[a]Built against Android 2.3 and tested on Android 4.2

[b]Dalvik VM is JDK 1.5 with some features from 1.6 (See Android documentation for details)

[c]Built against Android 5.0 and tested on Android 5.0.2.

[d]Dalvik VM is JDK 1.5 with some features from 1.6 (See Android documentation for details)

Make sure you are consistent in your use of static, dynamic, debug and release versions of the libraries. Do not link both static and dynamic libraries. Similarly, do not mix release and debug libraries.

Table 3.3 Running Instructions for Android Architectures provides details on the environment variables that must be set at run time for an Android architecture.

Table 3.4 Library-Creation Details for Android Architectures provides details on how the libraries were built. This table is provided strictly for informational purposes; you do not need to use these parameters to compile your application. You may find this information useful if you are involved in any in-depth debugging.

Connext DDS supports the Android operating system as a *target* platform. The target can be in one of two configurations: a consumer device (e.g., a Google™ Nexus™ 7 tablet) or as a "raw" Linux distribution. Building applications for the target occurs on a development machine using an Android SDK and, for C/C++, an Android NDK.

For a consumer device, all programs (applications and DDS utilities) must be installed on the device as Apps (***.apk** files). All Android Apps are loaded and executed by an instance of the Dalvik VM running as a Linux process. No Connext DDS components or libraries have to be pre-installed on the device—that is taken care of by the Android build and packaging tools. See the Android documentation for a full description of building and packaging Android Apps.

For a raw Linux distribution, all programs are executables that are linked with the necessary Connext DDS libraries (see Table 3.1 Supported Android Target Platforms). The build process is similar to other Linux variants, see Section 9.3 in the *RTI Connext DDS Core Libraries User's Manual*).

'Release' and 'Debug' Terminology:

Android and Connext DDS use these terms differently. For Android, "release" and "debug" refer to how application packages (***.apk**) are signed as part of the Android Security Model. A "release" package is cryptographically signed by a key that can be trusted by virtue of some certificate chain. A "debug" package is signed by a key distributed with the SDK. It says nothing about the origin of the package. It allows the package to be installed during development testing, hence "debug." For Connext DDS, debug means libraries created with debug symbols to facilitate debugging with gdb, for example. A "release" library does not contain debug information.

Additional Documentation:

See RTI Connext DDS Core Libraries Getting Started Guide Addendum for Android Systems

Table 3.2 Building Instructions for Android Architectures

API	Library Format	Required RTI Libraries and JAR Files[abc]	Required System Libraries	Required Compiler Flags
C++ (Traditional and Modern APIs)	Static Release	libnddscppz.a or libnddscpp2z.a libnddscz.a libnddscorez.a libgnustl_shared.a librticonnextmsgcppz.a	-L$(SYSROOT)/usr/lib -llog –lm -lc –lgnustl_shared	-march=armv7-a -mfloat-abi=softfp -mfpu=vfpv3-d16 -mlong-calls -DRTI_UNIX -DRTI_ANDROID
	Static Debug	libnddscppzd.a or libnddscpp2zd.a libnddsczd.a libnddscorezd.a libgnustl_shared.a librticonnextmsgcppzd.a		
	Dynamic Release	libnddscpp.so or libnddscpp2.so libnddsc.so libnddscore.so libgnustl_shared.so librticonnextmsgcpp.so		
	Dynamic Debug	libnddscppd.so or libnddscpp2d.so libnddscd.so libnddscored.so libgnustl_shared.so librticonnextmsgcppd.so		

[a]Choose libnddscpp*.* for the Traditional C++ API or libnddscpp2*.* for the Modern C++ API.

[b]The RTI C/C++/Java libraries are in $(NDDSHOME)/lib/<architecture>.

[c]The ***rticonnextmsg*** library only applies if you have the RTI Connext DDS Professional, Evaluation, or Basic package type. It is not provided with the RTI Connext DDS Core package type.

Table 3.2 Building Instructions for Android Architectures

API	Library Format	Required RTI Libraries and JAR Files[abc]	Required System Libraries	Required Compiler Flags
C	Static Release	libnddscz.a libnddscorez.a librticonnextmsgcz.a	-L$(SYSROOT)/usr/lib -llog –lm -lc	-march=armv7-a -mfloat-abi=softfp -mfpu=vfpv3-d16 -mlong-calls -DRTI_UNIX -DRTI_ANDROID
	Static Debug	libnddsczd.a libnddscorezd.a librticonnextmsgczd.a		
	Dynamic Release	libnddsc.so libnddscore.so librticonnextmsgc.so		
	Dynamic Debug	libnddscd.so libnddscored.so librticonnextmsgcd.so		
Java	Release	When not building Apps (*.apk): nddsjava.jar rticonnextmsg.jar When building Apps (*.apk): nddsjava.jar libnddsjava.so libnddsc.so libnddscore.so rticonnextmsg.jar	N/A	None required
	Debug	When not building Apps (*.apk): nddsjavad.jar rticonnextmsgd.jar When building Apps (*.apk): nddsjavad.jar libnddsjavad.so libnddscd.so libnddscored.so rticonnextmsgd.jar		

[a]Choose libnddscpp*.* for the Traditional C++ API or libnddscpp2*.* for the Modern C++ API.

[b]The RTI C/C++/Java libraries are in $(NDDSHOME)/lib/<architecture>.

[c]The ***rticonnextmsg*** library only applies if you have the RTI Connext DDS Professional, Evaluation, or Basic package type. It is not provided with the RTI Connext DDS Core package type.

Table 3.3 Running Instructions for Android Architectures

RTI Architecture	Library Format	Required Environment Variables
armv7aAndroid2.3gcc4.8 armv7aAndroid5.0gcc4.9ndkr10e	App (*.apk)	None
	Static	None
	Dynamic	LD_LIBRARY_PATH=$LD_LIBRARY_PATH: <path-to-ndds-libs>
armv7aAndroid2.3gcc4.8 for Java armv7aAndroid5.0gcc4.9ndkr10e for Java	App (*.apk)	None
	Dex	LD_LIBRARY_PATH=$LD_LIBRARY_PATH: <path-to-ndds-libs> CLASSPATH=<path-to-dex>/classes.dex

Table 3.4 Library-Creation Details for Android Architectures

RTI Architecture	Library Format	Compiler Flags Used by RTI
armv7aAndroid2.3gcc4.8	Release	-fpic -DLINUX -O -Wall -Wno-unknown-pragmas -Wno-address -Wno-unused-but-set-variable -DPtrIntType=long -DCSREAL_IS_FLOAT -DCPU=cortex-a9 -DTARGET=\"armv7aAndroid2.3gcc4.8\" -DNDEBUG -c -Wp,-MD
	Debug	-fpic -DLINUX -O -Wall -Wno-unknown-pragmas -Wno-address -Wno-unused-but-set-variable -DPtrIntType=long -DCSREAL_IS_FLOAT -DCPU=cortex-a9 -DTARGET=\"armv7aAndroid2.3gcc4.8\" -c -Wp,-MD
armv7aAndroid2.3gcc4.8 for Java	Release	-target 1.4 -source 1.4
	Debug	-target 1.4 -source 1.4 -g

Table 3.4 Library-Creation Details for Android Architectures

RTI Architecture	Library Format	Compiler Flags Used by RTI
armv7aAndroid5.0gcc4.9ndkr10e	Release	-fpic -DLINUX -O -Wall -Wno-unknown-pragmas -Wno-address -Wno-unused-but-setvariable -DPtrIntType=long -DCSREAL_IS_FLOAT -DCPU=cortex-a9 -DTARGET=\"armv7aAndroid5.0gcc4.9ndkr10e\" -DNDEBUG -c -Wp,-MD
	Debug	-fpic -DLINUX -O -Wall -Wno-unknown-pragmas -Wno-address -Wno-unused-but-setvariable -DPtrIntType=long -DCSREAL_IS_FLOAT -DCPU=cortex-a9 -DTARGET=\"armv7aAndroid5.0gcc4.9ndkr10e\" -c -Wp,-MD
armv7aAndroid5.0gcc4.9ndkr10e for Java	Release	-target 1.4 -source 1.4
	Debug	-target 1.4 -source 1.4 -g

3.1 Support for Modern C++ API

Connext DDS provides two different C++ APIs, which we refer to as the "Traditional C++" and "Modern C++" APIs. The Modern C++ API is available for all Android platforms.

For more information on Modern C++, see "Traditional vs. Modern C++" in the User's Manual.

3.2 Multicast Support

Multicast is available on supported Android platforms and is configured out of the box. That is, the default value for the initial peers list (NDDS_DISCOVERY_PEERS) includes a multicast address. See the API Reference HTML documentation for more information. Multicast has not been tested for this release and so, though available, is not officially supported. This should be addressed in a future release.

3.3 Transports

- **Shared memory:** Not supported for this release. For a consumer device, shared memory communication between Apps is often not desirable.
- **UDPv4**: Supported and enabled by default.
- **UDPv6**: Not supported. The IPv6 stack implementation has been evolving in parallel with the Android OS. For many of the supported Android versions there is either no or insufficient IPv6 support.

- **TCP/IPv4**: Supported.
- Secure WAN Transport: Supported. (However, RTI WAN Server is not supported.)

3.4 Monotonic Clock Support

The monotonic clock (described in "Clock Selection" in the User's Manual) is supported on all Android platforms.

3.5 Thread Configuration

Table 3.5 Thread Settings for Android Platforms lists the thread settings for Android platforms.

Table 3.6 Thread-Priority Definitions for Android Platforms lists the thread-priority definitions for Android platforms.

Table 3.5 Thread Settings for Android Platforms

Applicable Threads	DDS_ ThreadSettings_ t	Platform-Specific Setting
Asynchronous Publisher, Asynchronous flushing thread	mask	OS default thread type
	priority	OS default thread priority
	stack_size	OS default stack size
	cpu_list	CPU core affinity not supported
	cpu_rotation	
Database thread	mask	DDS_THREAD_SETTINGS_STDIO
	priority	OS default thread priority
	stack_size	OS default stack size
	cpu_list	CPU core affinity not supported
	cpu_rotation	

Table 3.5 Thread Settings for Android Platforms

Applicable Threads	DDS_ ThreadSettings_ t	Platform-Specific Setting
Event thread	mask	DDS_THREAD_SETTINGS_STDIO | DDS_THREAD_ SETTINGS_FLOATING_POINT
	priority	OS default thread priority
	stack_size	OS default stack size
	cpu_list	CPU core affinity not supported
	cpu_rotation	
ReceiverPool threads	mask	DDS_THREAD_SETTINGS_STDIO | DDS_THREAD_ SETTINGS_FLOATING_POINT
	priority	OS default thread priority
	stack_size	OS default stack size
	cpu_list	CPU core affinity not supported
	cpu_rotation	

Table 3.6 Thread-Priority Definitions for Android Platforms

Thread-Priority Definition	Operating-System Priority
THREAD_PRIORITY_DEFAULT	If any of these constants are used to set the priority of the thread in the QoS, the OS's default thread priority will be used.
THREAD_PRIORITY_HIGH	
THREAD_PRIORITY_ABOVE_NORMAL	
THREAD_PRIORITY_NORMAL	
THREAD_PRIORITY_BELOW_NORMAL	
THREAD_PRIORITY_LOW	

3.5.1 Support for Controlling CPU Core Affinity for RTI Threads

Support for controlling CPU core affinity (described in "Controlling CPU Core Affinity" in the User's Manual) is not available for Android platforms.

3.6 Durable Writer History and Durable Reader State Features

The Durable Writer History and Durable Reader State features are not supported on Android platforms.

3.7 Distributed Logger Support

RTI Distributed Logger is not supported on Android platforms.

3.8 Libraries Required for Using Monitoring

Make sure you are consistent in your use of static, dynamic, debug and release versions of the libraries. For example, if your Connext DDS application is linked with the static release version of the Connext DDS libraries, you will need to also use the static release version of the monitoring library. Do not link both static and dynamic libraries. Similarly, do not mix release and debug libraries.

If you are statically linking your application with DDS libraries and you want to add monitoring to your application, you will also need to statically link the monitoring library. The library cannot be loaded dynamically strictly through the QoS profile because it also depends on DDS to publish its data. Therefore, it depends on DDS; the DDS functionality would cause duplicate symbols to be found resulting, in the termination of the process.

Note: If you plan to use *static* libraries, the RTI library from Table 3.7 Additional Libraries for Using Monitoring must appear *first* in the list of libraries to be linked.

Table 3.7 Additional Libraries for Using Monitoring

Static Release	Static Debug	Dynamic Release	Dynamic Debug
librtimonitoringz.a	librtimonitoringzd.a	librtimonitoring.so	librtimonitoringd.so

3.9 Libraries Required for Using RTI Secure WAN Transport APIs

RTI Secure WAN Transport is only available on specific architectures. See the *RTI Secure WAN Transport Release Note*s and *RTI Secure WAN Transport Installation Guide* for details.

To use the Secure WAN Transport APIs, link against the additional libraries in Table 3.8 Additional Libraries for Using RTI Secure WAN Transport APIs on Android Systems. (Select the files appropriate for your chosen library format.)

Table 3.8 Additional Libraries for Using RTI Secure WAN Transport APIs on Android Systems

Library Format	RTI Secure WAN Transport Libraries[a]	OpenSSL Libraries[b]
Dynamic Release	libnddstransportwan.so libnddstransporttls.so	librtisslsupport.so
Dynamic Debug	libnddstransportwand.so libnddstransporttlsd.so	
Static Release	libnddstransportwanz.a libnddstransportlsz.a	
Static Debug	libnddstransportwanzd.a lib nddstransporttlszd.a	

3.10 Libraries Required for Using RTI TCP Transport APIs

To use the TCP Transport APIs, link against the additional libraries in Table 3.9 Additional Libraries for using RTI TCP Transport APIs on Android Systems. (Select the files appropriate for your chosen library format.)

Table 3.9 Additional Libraries for using RTI TCP Transport APIs on Android Systems

Library Format	RTI TCP Transport Libraries[c]	OpenSSL Libraries[d]
Dynamic Release	libnddstransporttcp.so	librtisslsupport.so
Dynamic Debug	libnddstransporttcpd.so	
Static Release	libnddstransporttcpz.a	
Static Debug	libnddstransporttcpzd.a	

[a]The libraries are in <NDDSHOME>/lib/<architecture>.

[b]These libraries are in <openssl install dir>/<architecture>/lib.

[c]These libraries are in <NDDSHOME>/lib/<architecture>.

[d]These libraries are in <openssl install dir>/<architecture>/lib.

Chapter 4 INTEGRITY Platforms

Table 4.1 Supported INTEGRITY Target Platforms lists the architectures supported on the INTEGRITY® operating system[a].

Table 4.1 Supported INTEGRITY Target Platforms

Operating System	CPU	Compiler	IP Stack	RTI Architecture Abbreviation
INTEGRITY 5.0.11	PPC 85XX	Multi 4.2.4	GHnet2 IP stack[b]	ppc85xxInty5.0.11.xes-p2020
INTEGRITY 10.0.2	x86	Multi 5.0.6	GHNet IPv4 stack	pentiumInty10.0.2.pcx86[c]
INTEGRITY 11.0.4	P4080	Multi 6.1	GHnet2 v2	p4080Inty11.devtree-fsl-e500mc.comp2012.1[d]
		Multi 6.1.4	GHNet2 v2	p4080Inty11.devtree-fsl-e500mc.comp2013.5.4[e]
	Pentium class	Multi 6.1.4	GHNet2	pentiumInty11.pcx86-smp

Table 4.2 Building Instructions for INTEGRITY Architectures lists the compiler flags and the libraries you will need to link into your application.

See also:

---

[a]For use with Windows and Solaris hosts, as supported by Green Hills Software.

[b]Kernel must be built using -lip4 or -lip46.

[c]See Required Patches for INTEGRITY 10.0.2 and 11.0.4 (Section 4.1 on page 30)

[d]See Required Patches for INTEGRITY 10.0.2 and 11.0.4 (Section 4.1 on page 30)

[e]See Required Patches for INTEGRITY 10.0.2 and 11.0.4 (Section 4.1 on page 30)

- Libraries Required for Using Distributed Logger (Section 4.8 on page 33)
- Libraries Required for Using Monitoring (Section 4.9 on page 34)

Do not mix release and debug libraries.

Table 4.3 Running Instructions for INTEGRITY Architectures provides details on the environment variables that must be set at run time for an INTEGRITY architecture.

Table 4.4 Library-Creation Details for INTEGRITY Architectures provides details on how the libraries were built. This table is provided strictly for informational purposes; you do not need to use these parameters to compile your application. You may find this information useful if you are involved in any in-depth debugging.

Table 4.2 Building Instructions for INTEGRITY Architectures

API	Library Format	Required RTI Libraries[abcd]	Required System Libraries[e]	Required Compiler Flags
C++ (Traditional and Modern APIs)	Static Release	libnddscppz.a or libnddscpp2z.a libnddscz.a libnddscorez.a librticonnextmsgcppz.a	libsocket.a libnet.a libposix.a	RTI_INTY --exceptions
	Static Debug	libnddscppzd.a or libnddscpp2zd.a libnddsczd.a libnddscorezd.a (libnddscppzd.dba or libnddscpp2zd.dba) (libnddsczd.dba) (libnddscorezd.dba) (librticonnextmsgczd.dba) librticonnextmsgcppzd.a		
C	Static Release	libnddscz.a libnddscorez.a librticonnextmsgcz.a		
	Static Debug	libnddsczd.a libnddscorezd.a (libnddsczd.dba) (libnddscorezd.dba) (librticonnextmsgczd.dba) librticonnextmsgczd.a		

[a]Choose libnddscpp*.* for the Traditional C++ API or libnddscpp2*.* for the Modern C++ API.

[b]The *.dba files contain the debugging information. You can link without these, as long as they are located in the same directory as the matching *d.a file (so that the MULTI® IDE can find the debug information).

[c]The RTI C/C++ libraries are in $(NDDSHOME)/lib/<architecture>.

[d]The ***rticonnextmsg*** library only applies if you have the RTI Connext DDS Professional, Evaluation, or Basic package type. It is not provided with the RTI Connext DDS Core package type.

[e]Transports (other than the default IP transport) such as StarFabric may require linking in additional libraries. For further details, see the API Reference HTML documentation or contact support@rti.com.

Table 4.3 Running Instructions for INTEGRITY Architectures

RTI Architecture	Required Environment Variables
All INTEGRITY architectures	None

Table 4.4 Library-Creation Details for INTEGRITY Architectures

RTI Architecture	Library Format	Compiler Flags Used by RTI
p4080Inty11.devtree-fsl-e500mc.comp2012.1	Static Release	-bsp=devtree-fsl-e500mc --prototype_warnings --unknown_pragma_silent --link_once_templates
	Static Debug	-bsp=devtree-fsl-e500mc --prototype_warnings --unknown_pragma_silent --link_once_templates -G
p4080Inty11.devtree-fsl-e500mc.comp2013.5.4	Static Release	-bsp=devtree-fsl-e500mc --prototype_warnings -non-shared --exceptions --unknown-pragma-silent --link_once_templates
	Static Debug	-bsp=devtree-fsl-e500mc --prototype_warnings -non-shared --exceptions
pentiumInty10.0.2.pcx86	Static Release	-bspname=pcx86 -prefixed_msgs --unknown_pragma_silent -G -DPtrIntType=long -DCSREAL_IS_FLOAT -DCPU= -DTARGET=\"pentiumInty10.0.2.pcx86\" -DNDEBUG -c
	Static Debug	-bspname=pcx86 -prefixed_msgs --unknown_pragma_silent -G -DPtrIntType=long -DCSREAL_IS_FLOAT -DCPU= -DTARGET=\"pentiumInty10.0.2.pcx86\" -c
pentiumInty11.pcx86-smp	Static Release	-bsp=pcx86-smp -prefixed_msgs --unknown_pragma_silent --link_once_templates -fexceptions -DPtrIntType=long -DCSREAL_IS_FLOAT -DCPU=pentium -DTARGET=\"pentiumInty11.pcx86-smp\" -DNDEBUG
	Static Debug	-bsp=pcx86-smp -prefixed_msgs --unknown_pragma_silent --link_once_templates -fexceptions -DRTS_INTY -DPtrIntType=long -DCSREAL_IS_FLOAT -DCPU=pentium -DTARGET=\"pentiumInty11.pcx86-smp\"
ppc85xxInty5.0.11.xes-p2020	Static Release	-bspname=xes-p2020 -prefixed_msgs --unknown_pragma_silent -G -DPtrIntType=long -DCSREAL_IS_FLOAT -DCPU= -DTARGET=\"ppc85xxInty5.0.11.xes-p2020\" -DNDEBUG -c
	Static Debug	-bspname=xes-p2020 -prefixed_msgs --unknown_pragma_silent -G -DPtrIntType=long -DCSREAL_IS_FLOAT -DCPU= -DTARGET=\"ppc85xxInty5.0.11.xes-p2020\" -c

4.1 Required Patches for INTEGRITY 10.0.2 and 11.0.4

For INTEGRITY 10.0.2 and 11.0.4 platforms, you must install these patches from Green Hills Software:

- INTEGRITY 10.0.2 Platforms
  - pentiumInty10.0.2.pcx86: **patch_6901.iff**
- INTEGRITY 11.0.4 Platforms
  - p4080Inty11.devtree-fsl-e500mc.comp2012.1: **patch_7584.iff** and **patch_7585.iff**
  - p4080Inty11.devtree-fsl-e500mc.comp2013.5.4: **patch_8154.iff**, **patch_8155.iff**, **patch_8246.iff**

For more information on these patches, please contact your Green Hills Software representative.

4.2 Support for Modern C++ API

Connext DDS provides two different C++ APIs, which we refer to as the "Traditional C++" and "Modern C++" APIs. The Modern C++ API is available for these INTEGRITY platforms:

- INTEGRITY 10.0.2 on an x86 CPU
- INTEGRITY 11.0.4

For more information on Modern C++, see "Traditional vs. Modern C++" in the User's Manual.

4.3 Multicast Support

Multicast is supported on all INTEGRITY platforms.

4.4 Supported Transports

Shared memory: Supported, enabled by default. To clean up shared memory resources, reboot the kernel.

UDPv4: Supported, enabled by default.

UDPv6: Not supported.

TCP/IPv4: Not supported.

4.5 Monotonic Clock Support

The monotonic clock (described in "Clock Selection" in the User's Manual) is not supported on INTEGRITY platforms.

4.6 Thread Configuration

Table 4.5 Thread Settings for INTEGRITY Platforms lists the thread settings for INTEGRITY platforms.

Table 4.6 Thread-Priority Definitions for INTEGRITY 5 and 11 Platforms and Table 4.7 Thread-Priority Definitions for INTEGRITY 10 Platforms list the thread-priority definitions.

Table 4.5 Thread Settings for INTEGRITY Platforms

Applicable Thread	DDS_ ThreadSettings_ t	Platform-Specific Setting
Asynchronous Publisher, Asynchronous flushing thread	mask	OS default thread type
	priority	127 for INTEGRITY 10.0.2 16 for all other supported INTEGRITY platforms
	stack_size	20*1024
	cpu_list	CPU core affinity not supported
	cpu_rotation	CPU core affinity not supported
Database thread	mask	DDS_THREAD_SETTINGS_STDIO
	priority	60 1 (INTEGRITY 10.0.2 only)
	stack_size	20*1024
	cpu_list	CPU core affinity not supported
	cpu_rotation	CPU core affinity not supported
Event thread	mask	DDS_THREAD_SETTINGS_STDIO | DDS_THREAD_ SETTINGS_FLOATING_POINT
	priority	80
	stack_size	4*20*1024
	cpu_list	CPU core affinity not supported
	cpu_rotation	CPU core affinity not supported

Table 4.5 Thread Settings for INTEGRITY Platforms

Applicable Thread	DDS_ ThreadSettings_ t	Platform-Specific Setting
ReceiverPool threads	mask	DDS_THREAD_SETTINGS_STDIO
	priority	100
	stack_size	4*20*1024
	cpu_list	CPU core affinity not supported
	cpu_rotation	CPU core affinity not supported

Table 4.6 Thread-Priority Definitions for INTEGRITY 5 and 11 Platforms

Thread-Priority Definition	Operating-System Priority
THREAD_PRIORITY_DEFAULT	16
THREAD_PRIORITY_HIGH	120
THREAD_PRIORITY_ABOVE_NORMAL	100
THREAD_PRIORITY_NORMAL	90
THREAD_PRIORITY_BELOW_NORMAL	80
THREAD_PRIORITY_LOW	60

Table 4.7 Thread-Priority Definitions for INTEGRITY 10 Platforms

Thread Priority Definitions	Operating System Priority
THREAD_PRIORITY_DEFAULT	127
THREAD_PRIORITY_HIGH	127
THREAD_PRIORITY_ABOVE_NORMAL	100
THREAD_PRIORITY_NORMAL	90

Table 4.7 Thread-Priority Definitions for INTEGRITY 10 Platforms

Thread Priority Definitions	Operating System Priority
THREAD_PRIORITY_BELOW_NORMAL	80
THREAD_PRIORITY_LOW	1

4.6.1 Socket-Enabled and POSIX-Enabled Threads are Required

On INTEGRITY platforms, Connext DDS internally relies on the POSIX API for many of its system calls. As a result, any thread calling Connext DDS must be POSIX-enabled. By default, the 'Initial' thread of an address space is POSIX-enabled, provided the address space has been linked with **libposix.a**. Additional user threads that call Connext DDS must be spawned from the Initial thread using **pthread_create**. Only then is the created thread also POSIX-enabled. Note that tasks created at build time using the Integrate utility are *not* POSIX-enabled.

Furthermore, threads calling Connext DDS must be socket-enabled. This can be achieved by calling **InitLibSocket()** before making any Connext DDS calls and calling **ShutdownLibSocket** before the thread terminates. Note that an Initial thread is, by default, socket-enabled when the address space is linked with **libsocket.a**. Please refer to the *INTEGRITY Development Guide* for more information.

4.6.2 Support for Controlling CPU Core Affinity for RTI Threads

Support for controlling CPU core affinity (described in "Controlling CPU Core Affinity" in the User's Manual) is not available for INTEGRITY platforms.

4.7 Durable Writer History and Durable Reader State Features

The Durable Writer History and Durable Reader State features are not supported on INTEGRITY platforms.

4.8 Libraries Required for Using Distributed Logger

RTI Distributed Logger is only supported for this architecture: p4080Inty11.devtree-fsl-e500m-c.comp2013.5.4. It is not supported on other INTEGRITY platforms.

Table 4.8 Additional Libraries for using RTI Distributed Logger lists the additional libraries you will need in order to use *Distributed Logger*.

Table 4.8 Additional Libraries for using RTI Distributed Logger

Language	Static	
	Release	Debug[a]
C	librtidlcz.a	librtidlczd.a (librtidlczd.dba)
C++ (Traditional API)	librtidlcz.a librtidlcppz.a	librtidlczd.a librtidlcppzd.a (librtidlczd.dba) (librtidlcppzd.dba)

4.9 Libraries Required for Using Monitoring

Make sure you are consistent in your use of debug and release versions of the libraries. For example, if your Connext DDS application is linked with the release version of the Connext DDS libraries, you will need to also use the release version of the monitoring library.

Note: The RTI library from Table 4.9 Additional Libraries for Using Monitoring must appear *first* in the list of libraries to be linked.

Table 4.9 Additional Libraries for Using Monitoring

Static Release	Static Debug
librtimonitoringz.a	librtimonitoringzd.a

4.10 Request-Reply Communication Pattern

The Connext DDS Professional, Research, Evaluation, and Basic packages include support for the Request-Reply Communication Pattern, for all platforms in Chapter 4 INTEGRITY Platforms and all programming languages, except as noted below.

When using C++, the following platform does not support the Request-Reply Communication Pattern:

- ppc85xxInty5.0.11.xes-p2020

[a]The *.dba files contain the debugging information. You can link without these, as long as they are located in the same directory as the matching *d.a file (so that the MULTI® IDE can find the debug information).

4.11 Diagnostics on INTEGRITY Systems

Connext DDS libraries for the INTEGRITY platforms use **consolestring()**, which prints debugging information to the serial console when available. Using the serial console as opposed to the target I/O window (host I/O) is generally recommended. Host I/O will affect the real-time performance of the target. For more information on **consolestring()**, please refer to the *INTEGRITY Development Guide.*

4.12 Running over IP Backplane on a Dy4 Champ-AVII Board

Connext DDS can run on all four CPUs, provided the following hold true:

- Connext DDS applications on CPUs B, C and D only exchange data with applications on a different CPU or off-board.
- The IP backplane and associated routing has been properly configured. Connext DDS has been tested with the following libraries built into the INTEGRITY kernel: **debug**, **res**, **load**, **socket**, **itcpip**, **lbp**, **queue**, **ifbp**, **idb**, **bsl**.

4.13 Multi-NIC Support on INTEGRITY 5.0

Due to limitations with the API of the InterPeak stack for INTEGRITY 5.0, Connext DDS only supports a single NIC when the InterPeak stack is used. This NIC must be called **"eth0**". By default on an INTEGRITY system, this will correspond to the first network card, which can be changed by reconfiguring the kernel. This limitation does not affect the InterNiche stack.

4.14 Out-of-the-box Transport Compatibility with Other Connext DDS Platforms

Due to some default kernel parameters on INTEGRITY platforms, the default value for **message_size_max** for the UDPv4 transport, and the default values for **message_size_max**, **received_message_count_max**, and **recv_buffer_size** for the shared-memory transport, are different than those for other platforms. This will cause out-of-the-box compatibility issues that may result in lack of communication. For more information on transport incompatibility, see Transport Compatibility, in the *RTI Connext DDS Core Libraries Release Notes*. The mismatch in transport configuration between INTEGRITY and other platforms applies to Connext DDS 5.1.0 and higher.

To address the compatibility issues, you can change the default transport settings of other platforms to match those of the INTEGRITY platform. Alternatively, you can update the INTEGRITY kernel parameters as described below so that the INTEGRITY platform will support larger transport settings.

The directive, GM_IP_FRAG_ENTRY_MAX_SIZE, limits the size of UDP packets that can be sent and received by INTEGRITY platforms. For details on this directive, please see Section 5.4.2 in the networking.pdf manual provided with the INTEGRITY kernel. The default value of GM_IP_FRAG_

ENTRY_MAX_SIZE is 9216 bytes (not 16,000 bytes as is stated in the INTEGRITY documentation), which is why the default **message_size_max** for all transports supported for INTEGRITY is 9216 bytes.

If you want to send UDP messages larger than 9k, you must increase the value of GM_IP_FRAG_ ENTRY_MAX_SIZE and rebuild the kernel. (You may also have to reconfigure other kernel parameters such as the socket, stack, and heap sizes to accommodate the larger value for GM_IP_FRAG_ENTRY_ MAX_SIZE.) Failing to increase this value will cause failures when sending large UDP packets, and in some cases (for example with the 5.0.11 kernel) the **sendto()** call will fail silently.

4.14.1 Smaller Shared-Memory Receive-Resource Queue Size

INTEGRITY's shared-memory pluggable transport uses the shared-memory POSIX API. This API is part of the standard INTEGRITY distribution and is shipped as a library. The current version (5.0.4) of this library uses a hard-coded value for the total amount of memory that can be shared with an address space. This limits the overall buffer space that can be used by the *DomainParticipants* within the same address space to communicate over shared memory with other *DomainParticipants*.

To allow more *DomainParticipants* to run within the same address space, we reduced the default size of the queue for each receive resource of the shared memory transport. The queue size is reduced to eight messages (the default for other platforms is 32). This change only applies to INTEGRITY architectures and this default value can be overwritten through the shared memory transport QoS.

4.14.2 Using Shared Memory on INTEGRITY Systems

Connext DDS uses the single address-space POSIX library to implement the shared-memory transport on INTEGRITY 10.0 operating systems.

To use shared-memory, you must configure your system to include the POSIX shared-memory library. The **posix_shm_manager** must be running in an "AddressSpace" solely dedicated to it. After building any Connext DDS application that uses shared memory, you must use the **intex** utility (provided with the INTEGRITY development environment) to pack the application with multiple address-spaces: one (or more) to contain the Connext DDS application(s), and another one to contain the **posix_shm_manager**.

Connext DDS will run on a target without the **posix_shm_manager**, but the POSIX functions will fail and return **ENOSYS**, and the participants will fail to communicate through shared memory.

To include the POSIX Shared-Memory Manager in its own Address Space:

The project files generated by *rtiddsgen* for MULTI will create the shared-memory manager for you. Please follow these steps:

1. Specify the path to your INTEGRITY distribution in the **_default.gpj** top-level project file by adding the following line (modify it according to the path to your INTEGRITY distribution):

```
-os_dir=/local/applications/integrity/integrity-10.0.2
```

2. Build the project.
3. Before running your Connext DDS application on a target, download the **posix_shm_manager** file (generated by the build) onto the target.

The POSIX Shared Memory Manager will start automatically after the download and your applications will be able to use shared memory.

Notes:

- Only *one* **posix_shm_manager** is needed on a particular target. INTEGRITY offers the option of building this **posix_shm_manager** *inside* the kernel. Please refer to the INTEGRITY documentation.
- If you are already using shared memory through the POSIX library, there may be a possible conflict.
- INTEGRITY 5 has two different types of POSIX library: a single-address space one (or 'light') and another one (complete POSIX implementation). Connext DDS uses the first one, but will work if you are using the complete POSIX implementation.

4.14.3 Shared Memory Limitations on INTEGRITY Systems

If several applications are running on the same INTEGRITY node and are using shared memory, once an application is stopped, it cannot be restarted. When the application is stopped (gracefully or ungracefully), any new application on the same domain index within the same DDS domain will fail to start until the shared memory manager is also restarted.

Additionally, if the application is stopped ungracefully, the remaining applications will print several error messages such as the following until Connext DDS purges the stopped application from its database:

```
Resource Manager send error = 0x9
```

This error message is logged from INTEGRITY's POSIX shared memory manager, *not* from Connext DDS. The error message is benign and will not prevent the remaining applications from communicating with each other or with application on other nodes.

The workaround is to either restart the stopped application with a different participant index or shut down all the other applications and the shared memory manager, then restart everything.

4.15 Using rtiddsping and rtiddsspy on PowerPC INTEGRITY Systems

While the RTI libraries for INTEGRITY can be used with any BSP, providing the PowerPC processor falls under the same category (for example, the ppc7400... RTI libraries can be used on any target with a PPC74xx processor), *rtiddsping* and *rtiddsspy* are provided as executables, and therefore are BSP-dependent. You will not be able to run them successfully on your target if it is not compatible with the BSP listed

in the architecture name (such as mvme5100-7400). Please refer to your hardware documentation for peripheral compatibility across BSPs.

4.16 Issues with INTEGRITY Systems

4.16.1 Delay When Writing to Unreachable Peers

On INTEGRITY systems, if a publishing application's initial peers list includes a nonexistent (or simply unreachable) host, calls to **write()** may block for approximately 1 second.

This long block is caused by the stack trying to resolve the invalid/unreachable host. Most IP stacks do not block the sending thread because of this reason, and you may include invalid/unreachable hosts in your initial-peers list. If you find that your stack does block the sending thread, please consult your IP stack vendor on how to change its behavior. [RTI Issue ID CORE-1637]

4.16.2 Linking with 'libivfs.a' without a File System

If you link your application with **libivfs.a** and are using a system that does not have a file system, you may notice the application blocks for 2 seconds at start-up.

4.16.3 Compiler Warnings Regarding Unrecognized #pragma Directives

Building Connext DDS projects for INTEGRITY causes the compiler to produce several warnings about #pragma directives not recognized in some Connext DDS header files. For example:

```
Building default.bld
"C:/ndds/ndds.4.4x/include/ndds/dds_c/dds_c_infrastructure.h", line 926:
warning: unrecognized #pragma
  #pragma warning(push)
          ^
"C:/ndds/ndds.4.4x/include/ndds/dds_c/dds_c_infrastructure.h", line 927:
warning: unrecognized #pragma
  #pragma warning(disable:4190)
          ^
"C:/ndds/ndds.4.4x/include/ndds/dds_c/dds_c_infrastructure.h", line 945:
warning: unrecognized #pragma
  #pragma warning(pop)
          ^
```

These warnings do not compromise the final application produced and can be safely ignored.

4.16.4 Warning when Loading Connext DDS Applications on INTEGRITY Systems

When a Connext DDS application compiled with the *rtiddsgen*-generated project files is loaded on an INTEGRITY 5.0.x target, the following warning appears:

```
"Warning: Program is linked with libc.so POSIX signals
and cancellation will not work."
```

The Connext DDS libraries do not use the additional features provided by the full POSIX implementation, therefore the warning can safely be ignored. This warning is due to the fact that the *rtiddsgen*-generated project files use the Single AddressSpace POSIX library by default, not the full POSIX implementation on INTEGRITY (POSIX System). The Connext DDS libraries only require Single AddressSpace POSIX to function correctly, but will still work if you are using the POSIX System. The message indicates that items such as inter-process signaling or process-shared semaphores will not be available (more information can be found in the *INTEGRITY Libraries and Utilities User's Guide*, chapter "Introduction to POSIX on INTEGRITY").

Chapter 5 iOS Platforms

Table 5.1 iOS Platforms lists the supported iOS architectures.

Table 5.1 iOS Platforms

Operating System	CPU	Compiler	RTI Architecture Abbreviation
iOS® 8.2	Dual-Core 64-bit Apple® A7	clang6.1	arm64iOS8clang6.1
	x86	clang 6.1	x86_64iOS8clang6.1

Table 5.2 Building Instructions for iOS Architectures lists the compiler flags and libraries you will need to link into your application. Make sure you are consistent in your use of release and debug versions of the libraries. Do not mix release and debug libraries.

Table 5.2 Building Instructions for iOS Architectures

API	Library Format	Required RTI Libraries [a] [b]	Required System Libraries	Required Compiler Flags
C	Static Release	libnddscz.a libnddscorez.a librticonnextmsgcz.a	For arm64iOS8clang6.1: -arch arm64 For x86_64iOS8clang6.1: -arch x86_64	-DRTI_UNIX -DRTI_IOS
	Static Debug	libnddsczd.a libnddscorezd.a librticonnextmsgczd.a		
C++ (Traditional and Modern APIs)	Static Release	libnddscppz.a or libnddscpp2z.a libnddscz.a libnddscorez.a librticonnextmsgcppz.a		
	Static Debug	libnddscppzd.a or libnddscpp2zd.a libnddsczd.a libnddscorezd.a librticonnextmsgcppzd.a		

Table 5.3 Running Instructions for iOS Architectures provides details on the environment variables that must be set at run time.

Table 5.3 Running Instructions for iOS Architectures

RTI Architecture	Library Format	Environment Variables
arm64iOS8clang6.1 x86_64iOS8clang6.1	Static	None required

Table 5.4 Library-Creation Details for iOS Architectures provides details on how the iOS libraries were built. This table is provided strictly for informational purposes; you do not need to use these parameters to

[a] Choose libnddscpp*.* for the Traditional C++ API or libnddscpp2*.* for the Modern C++ API

[b] The ***rticonnextmsg*** library only applies if you have the RTI Connext DDS Professional, Evaluation, or Basic package type. It is not provided with the RTI Connext DDS Core package type.

compile your application. You may find this information useful if you are involved in any in-depth debugging.

Table 5.4 Library-Creation Details for iOS Architectures

RTI Architecture	Library Format (Static)	Compiler Flags Used by RTI
arm64iOS8clang6.1	Release	arm64iOS8clang6.1 -arch arm64 -Wno-trigraphs -fpascal-strings -fasm-blocks -fmessage-length=0 -fdiagnostics-show-note-include-stack -fmacro-backtrace-limit=0 -O0 -Wparentheses -Wswitch -Wno-unknown-pragmas -Wno-shadow -Wno-four-char-constants -Wno-conversion -Wno-constant-conversion -Wno-int-conversion -Wno-bool-conversion -Wno-enum-conversion -Wshorten-64-to-32 -Wpointer-sign -Wno-newline-eof -Wno-return-type-c-linkage -Wno-c+11-narrowing -stdlib=libc+ -std=c++11
	Debug	
x86_64iOS8clang6.1	Release	x86_64iOS8clang6.1 -arch x86_64 -Wno-trigraphs -fpascal-strings -fasm-blocks -fmessage-length=0 -fdiagnostics-show-note-include-stack -fmacro-backtrace-limit=0 -O0 -Wparentheses -Wswitch -Wno-unknown-pragmas -Wno-shadow -Wno-four-char-constants -Wno-conversion -Wno-constant-conversion -Wno-int-conversion -Wno-bool-conversion -Wno-enum-conversion -Wshorten-64-to-32 -Wpointer-sign -Wno-newline-eof -Wno-return-type-c-linkage -Wno-c+11-narrowing -stdlib=libc+ -std=c++11
	Debug	

5.1 Supported Languages

The iOS libraries support the C, C++, C++03, and C++11 APIs.

5.1.1 Support for Modern C++ API

Connext DDS provides two different C++ APIs, which we refer to as the "Traditional C++" and "Modern C++" APIs. The Modern C++ API is available for all supported iOS platforms.

For more information on Modern C++, see "Traditional vs. Modern C++" in the User's Manual.

5.2 Multicast Support

Multicast is supported on all iOS platforms and is configured out of the box. That is, the default value for the initial peers list (**NDDS_DISCOVERY_PEERS**) includes a multicast address. See the API Reference HTML documentation for more information.

5.3 Transports

- **Shared memory:** Not supported.
- **UDPv4:** Supported and enabled by default.

- **UDPv6:**Not supported.
- **TCP/IPv4:** Supported.
- **Secure WAN Transport:** Supported. (However, RTI WAN Server is not supported.)

5.4 Unsupported Features

These features are not supported for iOS platforms:

- Controlling CPU Core Affinity
- Monotonic clock
- Durable Writer History and Durable Reader State

5.5 Thread Configuration

See Table 5.5 Thread Settings for iOS Platforms and Table 5.6 Thread-Priority Definitions for iOS Platforms.

5.5.1 Support for Controlling CPU Core Affinity for RTI Threads

Support for controlling CPU core affinity (described in "Controlling CPU Core Affinity" in the User's Manual) is not available for iOS platforms.

Table 5.5 Thread Settings for iOS Platforms

Applicable Thread	DDS_ ThreadSettings_ t	Platform-Specific Setting
Asynchronous Publisher, Asynchronous flushing thread	mask	OS default thread type
	priority	OS default thread priority
	stack_size	OS default thread stack size
	cpu_list	CPU core affinity not supported
	cpu_rotation	CPU core affinity not supported

Table 5.5 Thread Settings for iOS Platforms

Applicable Thread	DDS_ ThreadSettings_ t	Platform-Specific Setting
Database thread	mask	DDS_THREAD_SETTINGS_STDIO
	priority	OS default thread priority
	stack_size	OS default thread stack size
	cpu_list	CPU core affinity not supported
	cpu_rotation	CPU core affinity not supported
Event thread	mask	DDS_THREAD_SETTINGS_STDIO | DDS_THREAD_ SETTINGS_FLOATING_POINT
	priority	OS default thread priority
	stack_size	OS default thread stack size
	cpu_list	CPU core affinity not supported
	cpu_rotation	CPU core affinity not supported
ReceiverPool threads	mask	DDS_THREAD_SETTINGS_STDIO | DDS_THREAD_ SETTINGS_FLOATING_POINT
	priority	OS default thread priority
	stack_size	OS default thread stack size
	cpu_list	CPU core affinity not supported
	cpu_rotation	CPU core affinity not supported

Table 5.6 Thread-Priority Definitions for iOS Platforms

Thread-Priority Definition	Operating-System Priority
THREAD_PRIORITY_DEFAULT	If any of these constants are used to set the priority of the thread in the QoS, the OS's default thread priority will be used.
THREAD_PRIORITY_HIGH	
THREAD_PRIORITY_ABOVE_NORMAL	
THREAD_PRIORITY_NORMAL	
THREAD_PRIORITY_BELOW_NORMAL	
THREAD_PRIORITY_LOW	

5.6 Libraries Required for Using Distributed Logger

RTI Distributed Logger is supported on the platforms in Table 5.1 iOS Platforms.

To use the Distributed Logger APIs, link against the additional libraries in Table 5.7 Additional Libraries for using RTI Distributed Logger .

Table 5.7 Additional Libraries for using RTI Distributed Logger

Language	Release	Debug
C	librtidlcz.a	librtidlczd.a
C++ (Traditional and Modern APIs)	librtidlcppz.a	librtidlcppzd.a

5.7 Libraries Required for Using Monitoring

Make sure you are consistent in your use of debug and release versions of the libraries. For example, if your Connext DDS application is linked with the release version of the Connext DDS libraries, you will need to also use the release version of the monitoring library. Do not mix release and debug libraries.

Note: The RTI library from Table 5.8 Additional Libraries for Using Monitoring must appear *first* in the list of libraries to be linked.

Table 5.8 Additional Libraries for Using Monitoring

Static Release	Static Debug
librtimonitoringz.a	librtimonitoringzd.a

5.8 Libraries Required for Using RTI Secure WAN Transport

To use RTI Secure WAN Transport,. see the RTI Secure WAN Transport Release Notes and RTI Secure WAN Transport Installation Guide for details.

To use the Secure WAN Transport APIs, link against the additional libraries in Table 5.9 Additional Libraries for using RTI Secure WAN Transport APIs on iOS Systems. (Select the files appropriate for your chosen library format.)

Table 5.9 Additional Libraries for using RTI Secure WAN Transport APIs on iOS Systems

Library Format	RTI Secure WAN Transport Libraries[a]	OpenSSL Libraries[b]
Static Release	libnddstransportwanz.a libnddstransporttlsz.a	libsslz.a libcryptoz.a
Static Debug	libnddstransportwanzd.a libnddstransporttlszd.a	

[a]The libraries are in <NDDSHOME>/lib/<*architecture*>.

[b]These libraries are in <*openssl install dir*>/<*architecture*>/lib.

Chapter 6 Linux Platforms

First, see the basic instructions for compiling on Linux platforms provided in "Building Applications" in the User's Manual. The following tables provide supplemental information.

Table 6.1 Linux Platforms on ARM CPUs through Table 6.3 Linux Platforms on PowerPC CPUs list the supported Linux architectures.

Table 6.1 Linux Platforms on ARM CPUs

Operating System	CPU	Compiler	RTI Architecture Abbreviation
Raspbian Wheezy 7.0 (3.x kernel)	ARMv6	gcc 4.7.2[a]	armv6vfphLinux3.xgcc4.7.2
		Java Platform, Standard Edition JDK 1.7 or 1.8	

[a]Requires Linaro Gnueabihf Cross Compiler

Table 6.2 Linux Platforms on Intel CPUs

Operating System	CPU	Compiler	RTI Architecture Abbreviation
CentOS 5.4, 5.5 (2.6 kernel)	x86	gcc 4.1.2	i86Linux2.6gcc4.1.2
		Java Platform, Standard Edition JDK 1.7 or 1.8	
	x64	gcc 4.1.2	x64Linux2.6gcc4.1.2
		Java Platform, Standard Edition JDK 1.7 or 1.8	
CentOS 6.0, 6.2-6.4 (2.6 kernel)	x86	gcc 4.4.5	i86Linux2.6gcc4.4.5
		Java Platform, Standard Edition JDK 1.7 or 1.8	
	x64	gcc 4.4.5	x64Linux2.6gcc4.4.5
		Java Platform, Standard Edition JDK 1.7 or 1.8	
CentOS 7.0 (3.x kernel)	x86	gcc 4.8.2	i86Linux3gcc4.8.2
		Java Platform, Standard Edition JDK 1.7 or 1.8	
	x64	gcc 4.8.2	x64Linux3gcc4.8.2
		Java Platform, Standard Edition JDK 1.7 or 1.8	
Red Hat Enterprise Linux 4.0[a] (2.6 kernel)	x86	gcc 3.4.3	i86Linux2.6gxx3.4.3x
	x64	gcc 3.4.3	64Linux2.6gcc3.4.4

[a]Running *rtiddsgen* on this platform is not supported because it uses an older JRE. You can, however, run *rtiddsgen* on a newer Linux platform to generate code that can be used on the Red Hat Enterprise Linux 4.0 target.

Table 6.2 Linux Platforms on Intel CPUs

Operating System	CPU	Compiler	RTI Architecture Abbreviation
Red Hat Enterprise Linux 5.0 (2.6 kernel)	x86	gcc 4.1.1	i86Linux2.6gcc4.1.1
		Java Platform, Standard Edition JDK 1.7 or 1.8	
	x64	gcc 4.1.1	x64Linux2.6gcc4.1.1
		Java Platform, Standard Edition JDK 1.7 or 1.8	
Red Hat Enterprise Linux 5.1, 5.2, 5.4, 5.5 (2.6 kernel)	x86	gcc 4.1.2	i86Linux2.6gcc4.1.2
		Java Platform, Standard Edition JDK 1.7 or 1.8	
	x64	gcc 4.1.2	x64Linux2.6gcc4.1.2
		Java Platform, Standard Edition JDK 1.7 or 1.8	
Red Hat Enterprise Linux 5.2 with Real-Time Extensions (2.6 kernel)	x86	gcc 4.1.2	i86Linux2.6gcc4.1.2
		Java Platform, Standard Edition JDK 1.7 or 1.8	
Red Hat Enterprise Linux 6.0-6.5, 6.7 (2.6 kernel)	x86	gcc 4.4.5	i86Linux2.6gcc4.4.5
		Java Platform, Standard Edition JDK 1.7 or 1.8	
	x64	gcc 4.4.5	x64Linux2.6gcc4.4.5
		Java Platform, Standard Edition JDK 1.7 or 1.8	
Red Hat Enterprise Linux7.0 (3.x kernel)	x86	gcc 4.8.2	i86Linux3gcc4.8.2
		Java Platform, Standard Edition JDK 1.7 or 1.8	
	x64	gcc 4.8.2	x64Linux3gcc4.8.2
		Java Platform, Standard Edition JDK 1.7 or 1.8	

Table 6.2 Linux Platforms on Intel CPUs

Operating System	CPU	Compiler	RTI Architecture Abbreviation
SUSE Linux Enterprise Server 11 SP2 (3.x kernel)	x86	gcc 4.3.4	i86Linux3gcc4.3.4
		Java Platform, Standard Edition JDK 1.7 or 1.8	
SUSE Linux Enterprise Server 11 SP2, SP3 (2.6 kernel)	x64	gcc 4.3.4	x64Linux2.6gcc4.3.4
		Java Platform, Standard Edition JDK 1.7 or 1.8	
Ubuntu Server 12.04 LTS	x86	gcc 4.6.3	i86Linux3.xgcc4.4.3
		Java Platform, Standard Edition JDK 1.7 or 1.8	
	x64	gcc 4.6.3	x64Linux3.xgcc4.6.3
		Java Platform, Standard Edition JDK 1.7 or 1.8	
Ubuntu 14.04 LTS	x86	gcc 4.8.2	i86Linux3gcc4.8.2
		Java Platform, Standard Edition JDK 1.7 or 1.8	
	x64	gcc 4.8.2	x64Linux3gcc4.8.2
		Java Platform, Standard Edition JDK 1.7 or 1.8	
Wind River Linux 4 (2.6 kernel)	x64	gcc 4.4.1	x64WRLinux2.6gcc4.4.1

Table 6.3 Linux Platforms on PowerPC CPUs

Operating System	CPU	Compiler	RTI Architecture Abbreviation
Freescale P2020RDB (2.6.32 kernel)	PPC 85xx	Freescale gcc.4.3.74 based on gcc.4.3.2	ppc85xxLinux2.6gcc4.3.2
Wind River Linux 3	PPC 85xx	gcc 4.3.2	ppc85xxWRLinux2.6gcc4.3.2
Yellow Dog® Linux 4.0 (2.6 kernel)	PPC 74xx (such as 7410)	gcc 3.3.3	ppc7400Linux2.6gcc3.3.3

Table 6.4 Building Instructions for Linux Architectures lists the compiler flags and libraries you will need to link into your application.

See also:

- Libraries Required for Using Distributed Logger (Section 6.7 on page 63)
- Libraries Required for Using Monitoring (Section 6.8 on page 64)
- Libraries Required for Using RTI Secure WAN Transport APIs (Section 6.9 on page 64)
- Libraries Required for Using RTI TCP Transport APIs (Section 6.10 on page 65)

Make sure you are consistent in your use of static, dynamic, debug and release versions of the libraries. Do not link both static and dynamic libraries. Similarly, do not mix release and debug libraries.

Table 6.4 Building Instructions for Linux Architectures

API	Library Format	Required RTI Libraries or Jar Files[abc]	Required System Libraries	Required Compiler Flags
C++ (Traditional and Modern APIs)	Static Release	libnddscppz.a or libnddscpp2z.a libnddscz.a libnddscorez.a librticonnextmsgcz.a	All *Linux2.6gcc3* architectures: -ldl -lnsl -lm -L/usr/lib/nptl -lpthread -lrt All other Linux architectures: -ldl -lnsl -lm -lpthread -lrt	64-bit architectures: -DRTI_UNIX -m64 32-bit architectures: -DRTI_UNIX -m32 For i86Linux3gcc4.8.2 and x64Linux3gcc4.8.2 when running on Ubuntu CPU for dynamic release and dynamic debug libraries, also use the following: -Wl,--no-as-needed
	Static Debug	libnddscppzd.a or libnddscpp2zd.a libnddsczd.a libnddscorezd.a librticonnextmsgcppzd.a		
	Dynamic Release	libnddscpp.so or libnddscpp2.so libnddsc.so libnddscore.so librticonnextmsgcpp.so		
	Dynamic Debug	libnddscppd.so or libnddscpp2d.so libnddscd.so libnddscored.so librticonnextmsgcppd.so		

[a]Choose libnddscpp*.* for the Traditional C++ API or libnddscpp2*.* for the Modern C++ API.

[b]RTI C/C++/Java libraries are in <NDDSHOME>/lib/<architecture>. The jar files are in <NDDSHOME>/lib/java.

[c]The ***rticonnextmsg*** library only applies if you have the RTI Connext DDS Professional, Evaluation, or Basic package type. It is not provided with the RTI Connext DDS Core package type.

Table 6.4 Building Instructions for Linux Architectures

API	Library Format	Required RTI Libraries or Jar Files[abc]	Required System Libraries	Required Compiler Flags
C	Static Release	libnddscz.a libnddscorez.a librticonnextmsgcz.a	All *Linux2.6gcc3* architectures: -ldl -lnsl -lm -L/usr/lib/nptl -lpthread -lrt All other Linux architectures: -ldl -lnsl -lm -lpthread -lrt	64-bit architectures: -DRTI_UNIX -m64 32-bit architectures: -DRTI_UNIX -m32 For i86Linux3gcc4.8.2 and x64Linux3gcc4.8.2 when running on Ubuntu CPU for dynamic release and dynamic debug libraries, also use the following: -Wl,--no-as-needed
	Static Debug	libnddsczd.a libnddscorezd.a librticonnextmsgczd.a		
	Dynamic Release	libnddsc.so libnddscore.so librticonnextmsgc.so		
	Dynamic Debug	libnddscd.so libnddscored.so librticonnextmsgcd.so		
Java	Release	nddsjava.jar rticonnextmsg.jar	N/A	None required
	Debug	nddsjavad.jar rticonnextmsgd.jar		

Table 6.5 Running Instructions for Linux Architectures provides details on the environment variables that must be set at run time for a Linux architecture. When running on 64-bit Java architectures (x64Linux2.6...), use the **-d64** flag on the command-line.

[a]Choose libnddscpp*.* for the Traditional C++ API or libnddscpp2*.* for the Modern C++ API.

[b]RTI C/C++/Java libraries are in <NDDSHOME>/lib/<architecture>. The jar files are in <NDDSHOME>/lib/java.

[c]The ***rticonnextmsg*** library only applies if you have the RTI Connext DDS Professional, Evaluation, or Basic package type. It is not provided with the RTI Connext DDS Core package type.

Table 6.5 Running Instructions for Linux Architectures

RTI Architecture	Library Format	Environment Variables
All supported Linux/SUSE architectures when using Java	N/A	LD_LIBRARY_PATH= ${NDDSHOME}/lib/<*architecture*>:${LD_LIBRARY_PATH} **Note:** For all 64-bit Java architectures (...64Linux...), use **-d64** in the command line.
All other supported Linux/SUSE architectures when not using Java	Static (Release & Debug)	None required
	Dynamic (Release & Debug)	LD_LIBRARY_PATH= ${NDDSHOME}/lib/<*architecture*>:${LD_LIBRARY_PATH}

Table 6.6 Library-Creation Details for Linux Architectures provides details on how the Linux libraries were built. This table is provided strictly for informational purposes; you do not need to use these parameters to compile your application. You may find this information useful if you are involved in any in-depth debugging.

Table 6.6 Library-Creation Details for Linux Architectures

RTI Architecture	Library Format (Static & Dynamic)	Compiler Flags Used by RTI
i86Linux2.6gcc3.4.3	Release	-fPIC -DLINUX -O -Wall -Wno-unknown-pragmas -DPtrIntType=long -DCSREAL_IS_FLOAT -DCPU=I80586 -DTARGET=\"i86Linux2.6gcc3.4.3\" -fmessage-length=0 -DNDEBUG -c -Wp,-MD
	Debug	-fPIC -DLINi86Linux2.6gcc3.4.3UX -g -Wall -Wno-unknown-pragmas -DPtrIntType=long -DCSREAL_IS_FLOAT -DCPU=I80586 -DTARGET=\"i86Linux2.6gcc3.4.3\" -fmessage-length=0 -c -Wp,-MD
i86Linux2.6gcc4.1.1	Release	-fPIC -DLINUX -O -Wall -Wno-unknown-pragmas -DPtrIntType=long -DCSREAL_IS_FLOAT -DCPU=I80586 -DTARGET=\"i86Linux2.6gcc4.1.1\" -fmessage-length=0 -DNDEBUG -c -Wp,-MD
	Debug	-fPIC -DLINUX -g -Wall -Wno-unknown-pragmas -DPtrIntType=long -DCSREAL_IS_FLOAT -DCPU=I80586 -DTARGET=\"i86Linux2.6gcc4.1.1\" -fmessage-length=0 -c -Wp,-MD

Table 6.6 Library-Creation Details for Linux Architectures

RTI Architecture	Library Format (Static & Dynamic)	Compiler Flags Used by RTI
i86Linux2.6gcc4.1.2	Release	-fPIC -DLINUX -O -Wall -Wno-unknown-pragmas -DPtrIntType=long -DCSREAL_IS_FLOAT -DCPU=I80586 -DTARGET=\"i86Linux2.6gcc4.1.2\" -fmessage-length=0 -DNDEBUG -c -Wp,-MD
	Debug	-fPIC -DLINUX -g -Wall -Wno-unknown-pragmas -DPtrIntType=long -DCSREAL_IS_FLOAT -DCPU=I80586 -DTARGET=\"i86Linux2.6gcc4.1.2\" -fmessage-length=0 -c -Wp,-MD
i86Linux2.6gcc4.4.5	Release	gcc -m32 -fPIC -DLINUX -O -Wall -Wno-unknown-pragmas -DPtrIntType=long -DCSREAL_IS_FLOAT -DCPU=I80586 -DTARGET=\\"i86Linux2.6gcc4.4.5\\" -DNDEBUG -Wp,-MD
	Debug	gcc -m32 -fPIC -DLINUX -g -Wall -Wno-unknown-pragmas -DPtrIntType=long -DCSREAL_IS_FLOAT -DCPU=I80586 -DTARGET=\\"i86Linux2.6gcc4.4.5\\" -Wp,-MD
i86Linux3gcc4.3.4	Release	-fPIC -DLINUX -O -Wall -Wno-unknown-pragmas -DPtrIntType=long -DCSREAL_IS_FLOAT -DCPU=I80586 -DTARGET=\"i86Linux3gcc4.3.4\" -fmessage-length=0 -DNDEBUG -c -Wp,-MD
	Debug	-fPIC -DLINUX -g -Wall -Wno-unknown-pragmas -DPtrIntType=long -DCSREAL_IS_FLOAT -DCPU=I80586 -DTARGET=\"i86Linux3gcc4.3.4\" -fmessage-length=0 -c-Wp,-MD
i86Linux3gcc4.8.2	Release	-m32 -fPIC -DLINUX -DRTI_LINUX26 -DRTI_LINUX -DRTI_POSIX_THREADS -DRTI_POSIX_SEMAPHORES -DRTI_CPU_AFFINITY -O -Wall -Wno-unknown-pragmas -DPtrIntType=long -DCSREAL_IS_FLOAT -DCPU=i686 -DRTI_ENDIAN_LITTLE -DRTI_THREADS -DRTI_MULTICAST -DRTI_SHARED_MEMORY -DRTI_IPV6 -DTARGET=\"i86Linux3gcc4.8.2\" -DNDEBUG -c -Wp,-MD
	Debug	-g -m32 -fPIC -DLINUX -DRTI_LINUX26 -DRTI_LINUX -DRTI_POSIX_THREADS -DRTI_POSIX_SEMAPHORES -DRTI_CPU_AFFINITY -O -Wall -Wno-unknown-pragmas -DPtrIntType=long -DCSREAL_IS_FLOAT -DCPU=i686 -DRTI_ENDIAN_LITTLE -DRTI_THREADS -DRTI_MULTICAST -DRTI_SHARED_MEMORY -DRTI_IPV6 -DTARGET=\"i86Linux3gcc4.8.2\" -DDEBUG -c -Wp,-MD

Table 6.6 Library-Creation Details for Linux Architectures

RTI Architecture	Library Format (Static & Dynamic)	Compiler Flags Used by RTI
ppc7400Linux2.6gcc3.3.3 [a]	Release	-fPIC -DLINUX -O -Wall -Wno-unknown-pragmas -DPtrIntType=long -DCSREAL_IS_FLOAT -DCPU=PPC7400 -DTARGET=\"ppc7400Linux2.6gcc3.3.3\" -DNDEBUG -c -Wp,-MD
	Debug	-fPIC -DLINUX -g -Wall -Wno-unknown-pragmas -DPtrIntType=long -DCSREAL_IS_FLOAT -DCPU=PPC7400 -DTARGET=\"ppc7400Linux2.6gcc3.3.3\" -c -Wp,-MD
ppc85xxLinux2.6gcc4.3.2	Release	-fPIC -DLINUX -g -Wall -Wno-unknown-pragmas -DPtrIntyType=long -DCREAL_IS_FLOAT -DCPU=e500 -DTARGET=\"ppc85xxLinux2.6gcc4.3.2\" -DNDEBUG -c -Wp, -MD
	Debug	-fPIC -DLINUX -g -Wall -Wno-unknown-pragmas -DPtrIntyType=long -DCREAL_IS_FLOAT -DCPU=e500 -DTARGET=\"ppc85xxLinux2.6gcc4.3.2\" -c -Wp, -MD
ppc85xxWRLinux2.6gcc4.3.2	Release	-mcpu=powerpc -msoft-float -fPIC -DLINUX -O -Wall -Wno-unknown-pragmas -DPtrIntType=long -DCSREAL_IS_FLOAT -DCPU=PPC32 -DTARGET=\"ppc85xxWRLinux2.6gcc4.3.2\" -DNDEBUG -Wp,-MD
	Debug	powerpc-wrs-linux-gnu-gcc -mcpu=powerpc -msoft-float -fPIC -DLINUX -g -Wall -Wno-unknown-pragmas -DPtrIntType=long -DCSREAL_IS_FLOAT -DCPU=PPC32 -DTARGET=\"ppc85xxWRLinux2.6gcc4.3.2\" -Wp,-MD
x64Linux2.6gcc4.1.1	Release	-m64 -fPIC -DLINUX -O -Wall -Wno-unknown-pragmas -DPtrIntType=long -DCSREAL_IS_FLOAT -DCPU=AMD64 -DTARGET=\"x64Linux2.6gcc4.1.1\" -DNDEBUG -c -Wp,-MD
	Debug	-m64 -fPIC -DLINUX -g -Wall -Wno-unknown-pragmas -DPtrIntType=long -DCSREAL_IS_FLOAT -DCPU=AMD64 -DTARGET=\"x64Linux2.6gcc4.1.1\" -fmessage-length=0 -c -Wp,-MD

[a]The C++ libnddscpp dynamic libraries were linked using g++; the C dynamic libraries, i.e., libnddscore and libnddsc, were linked using gcc.

Table 6.6 Library-Creation Details for Linux Architectures

RTI Architecture	Library Format (Static & Dynamic)	Compiler Flags Used by RTI
x64Linux2.6gcc4.1.2[a]	Release	-m64 -fPIC -DLINUX -O -Wall -Wno-unknown-pragmas -DPtrIntType=long -DCSREAL_IS_FLOAT -DCPU=AMD64 -DTARGET=\"x64Linux2.6gcc4.1.2\" -DNDEBUG -c -Wp,-MD
	Debug	-m64 -fPIC -DLINUX -g -Wall -Wno-unknown-pragmas -DPtrIntType=long -DCSREAL_IS_FLOAT -DCPU=AMD64 -DTARGET=\"x64Linux2.6gcc4.1.2\" -fmessage-length=0 -c -Wp,-MD
x64Linux2.6gcc4.3.4	Release	-m64 -fPIC -DLINUX -O -Wall -Wno-unknown-pragmas -DPtrIntType=long -DCSREAL_IS_FLOAT -DCPU=AMD64 -DTARGET=\"x64Linux2.6gcc\" -c -Wp,-MD
	Debug	-m64 -fPIC -DLINUX -g -Wall -Wno-unknown-pragmas -DPtrIntType=long -DCSREAL_IS_FLOAT -DCPU=AMD64 -DTARGET=\"x64Linux2.6gcc4.3.4\" -c -Wp,-MD
x64Linux2.6gcc4.4.5	Release	gcc -m64 -fPIC -DLINUX -O -Wall -Wno-unknown-pragmas -DPtrIntType=long -DCSREAL_IS_FLOAT -DCPU=AMD64 -DTARGET=\\"x64Linux2.6gcc4.4.5\\" -DNDEBUG -Wp,-MD
	Debug	gcc -m64 -fPIC -DLINUX -g -Wall -Wno-unknown-pragmas -DPtrIntType=long -DCSREAL_IS_FLOAT -DCPU=AMD64 -DTARGET=\\"x64Linux2.6gcc4.4.5\\" -Wp,-MD
x64Linux3gcc4.8.2	Release	-m64 -fPIC -DLINUX -DRTI_LINUX26 -DRTI_LINUX -DRTI_POSIX_THREADS -DRTI_POSIX_SEMAPHORES -DRTI_CPU_AFFINITY -O -Wall -Wno-unknown-pragmas -DPtrIntType=long -DCSREAL_IS_FLOAT -DCPU=i686 -DRTI_ENDIAN_LITTLE -DRTI_THREADS -DRTI_MULTICAST -DRTI_SHARED_MEMORY -DRTI_IPV6 -DTARGET=\"x64Linux3gcc4.8.2\" -DNDEBUG -c -Wp,-MD
	Debug	-g -m64 -fPIC -DLINUX -DRTI_LINUX26 -DRTI_LINUX -DRTI_POSIX_THREADS -DRTI_POSIX_SEMAPHORES -DRTI_CPU_AFFINITY -O -Wall -Wno-unknown-pragmas -DPtrIntType=long -DCSREAL_IS_FLOAT -DCPU=i686 -DRTI_ENDIAN_LITTLE -DRTI_THREADS -DRTI_MULTICAST -DRTI_SHARED_MEMORY -DRTI_IPV6 -DTARGET=\"x64Linux3gcc4.8.2\" -DDEBUG -c -Wp,-MD
x64WRLinux2.6gcc4.4.1	Release	-m64 -march=x86-64 -mtune=generic -fPIC -DLINUX -O -Wall -Wno-unknown-pragmas -DPtrIntType=long -DCSREAL_IS_FLOAT -DCPU=AMD64 -DNDEBUG
	Debug	-m64 -march=x86-64 -mtune=generic -fPIC -DLINUX -O -Wall -Wno-unknown-pragmas -DPtrIntType=long -DCSREAL_IS_FLOAT -DCPU=AMD64 -DNDEBUG

[a]The C++ libnddscpp dynamic libraries were linked using g++; the C dynamic libraries, i.e., libnddscore and libnddsc, were linked using gcc.

Table 6.6 Library-Creation Details for Linux Architectures

RTI Architecture	Library Format (Static & Dynamic)	Compiler Flags Used by RTI
All supported Linux architectures for Java	Dynamic Release	-target 1.4 -source 1.4
	Dynamic Debug	-target 1.4 -source 1.4 -g

6.1 Support for Modern C++ API

Connext DDS provides two different C++ APIs, which we refer to as the "Traditional C++" and "Modern C++" APIs. The Modern C++ API is available for all the platforms in Table 6.1 Linux Platforms on ARM CPUs through Table 6.3 Linux Platforms on PowerPC CPUs except Yellow Dog Linux 4.0 (ppc7400Linux2.6gcc3.3.3).

For more information on Modern C++, see "Traditional vs. Modern C++" in the User's Manual.

6.2 Multicast Support

Multicast is supported on all Linux platforms and is configured out of the box. That is, the default value for the initial peers list (**NDDS_DISCOVERY_PEERS**) includes a multicast address. See the API Reference HTML documentation for more information.

6.3 Supported Transports

Shared memory: Supported and enabled by default. To clean up shared memory resources, reboot the kernel.

UDPv4: Supported and enabled by default.

UDPv6: Supported for all platforms *except* Raspbian Wheezy 7.0 (armv6vfphLinux3.xgcc4.7.2).

The UDPv6 transport is not enabled by default, and the peers list must be modified to support IPv6.

Note: Traffic Class support is only provided on architectures with gcc 4.1.0 or later that support the UDPv6 transport.

TCP/IPv4: Supported on CentOS 5.4 and higher, Red Hat Enterprise Linux 4.0 and higher, Ubuntu server 10.04 and higher, Raspbian Wheezy 7.0, SUSE Linux Enterprise Server 11. (This is *not* a built-in transport.)

6.3.1 Shared Memory Support

To see a list of shared memory resources in use, please use the '**ipcs**' command. To clean up shared memory and shared semaphore resources, please use the '**ipcrm**' command.

The shared memory keys used by Connext DDS are in the range of 0x400000. For example:

```
ipcs -m | grep 0x004
```

The shared semaphore keys used by Connext DDS are in the range of 0x800000; the shared mutex keys are in the range of 0xb00000. For example:

```
ipcs -s | grep 0x008

ipcs -s | grep 0x00b
```

Please refer to the shared-memory transport online documentation for details on the shared memory and semaphore keys used by Connext DDS.

6.4 Monotonic Clock Support

The monotonic clock (described in "Clock Selection" in the User's Manual) is supported on platforms with all Linux 2.6 kernel or higher.

6.5 Thread Configuration

Table 6.7 Thread Settings for Linux Platforms lists the thread settings for Linux platforms.

Table 6.8 Thread-Priority Definitions for Linux Platforms and Table 6.9 Thread Kinds for Linux Platforms list the thread-priority definitions and thread kinds, respectively.

6.5.1 Native POSIX Thread Library (NPTL) Requirements

This section applies only to these platforms:

- **Red Hat Enterprise Linux 4.0:** i86Linux2.6gcc3.4.3
- **Yellow Dog Linux 4.0:** ppc7400Linux2.6gcc3.3.3

To use the above platforms, you must have the development version of Native POSIX Thread Library (NPTL) installed on your host system and the NPTL libraries on your target system.

- When you *build* the application, you must have the development NPTL library installed in **/usr/lib/nptl**. This library is not installed by default.
- To see if your system has NPTL installed, look for this directory: **/usr/lib/nptl**. It should contain these files: **libpthread.so** and **libpthread.a**.
- If NPTL is not installed, you will need to install a package that includes it, such as **nptl-devel**. This package is not typically part of a default installation. You can find it either in your original Linux installation media (CD/DVD) or, if you have upgraded your system, through the distribution's update site.
- When you *run* the application, it will automatically use the default NPTL library in **/lib/nptl**. You do not need the development library installed on the target system.

Make sure the environment variable LD_ASSUME_KERNEL is either not defined at all, or is set to 2.4.20 or higher. The middleware will not run if it is set to less than 2.4.20.[a]

6.5.2 Support for Controlling CPU Core Affinity for RTI Threads

Support for controlling CPU core affinity (described in "Controlling CPU Core Affinity" in the User's Manual) is available on all supported Linux/SUSE platforms.

Table 6.7 Thread Settings for Linux Platforms

Applicable Thread	DDS_ ThreadSettings_ t	Platform-Specific Setting
Asynchronous Publisher, Asynchronous flushing thread	mask	OS default thread type
	priority	OS default thread priority
	stack_size	OS default thread stack size
	cpu_list	Empty CPU list (Supported on Linux and SUSE platforms)
	cpu_rotation	DDS_THREAD_SETTINGS_CPU_NO_ROTATION (Supported on Linux and SUSE platforms)

[a]The dynamic loader (ld), is configured by default to load the NPTL library, as long as LD_ASSUME_KERNEL is NOT defined.

Table 6.7 Thread Settings for Linux Platforms

Applicable Thread	DDS_ThreadSettings_t	Platform-Specific Setting
Database thread	mask	DDS_THREAD_SETTINGS_STDIO
	priority	OS default thread priority
	stack_size	OS default thread stack size
	cpu_list	Empty CPU list (Supported on Linux and SUSE platforms)
	cpu_rotation	DDS_THREAD_SETTINGS_CPU_NO_ROTATION (Supported on Linux and SUSE platforms)
Event thread	mask	DDS_THREAD_SETTINGS_STDIO | DDS_THREAD_SETTINGS_FLOATING_POINT
	priority	OS default thread priority
	stack_size	OS default thread stack size
	cpu_list	Empty CPU list (Supported on Linux and SUSE platforms)
	cpu_rotation	DDS_THREAD_SETTINGS_CPU_NO_ROTATION (Supported on Linux and SUSE platforms)
ReceiverPool threads	mask	DDS_THREAD_SETTINGS_STDIO | DDS_THREAD_SETTINGS_FLOATING_POINT
	priority	OS default thread priority
	stack_size	OS default thread stack size
	cpu_list	Empty CPU list (Supported on Linux and SUSE platforms)
	cpu_rotation	DDS_THREAD_SETTINGS_CPU_NO_ROTATION (Supported on Linux and SUSE platforms)

Table 6.8 Thread-Priority Definitions for Linux Platforms

Thread-Priority Definition	Operating-System Priority
THREAD_PRIORITY_DEFAULT	If any of these constants are used to set the priority of the thread in the QoS, the OS's default thread priority will be used.
THREAD_PRIORITY_HIGH	
THREAD_PRIORITY_ABOVE_NORMAL	
THREAD_PRIORITY_NORMAL	
THREAD_PRIORITY_BELOW_NORMAL	
THREAD_PRIORITY_LOW	

Table 6.9 Thread Kinds for Linux Platforms

Thread Kinds	Operating-System Configuration[a]
DDS_THREAD_SETTINGS_FLOATING_POINT	N/A
DDS_THREAD_SETTINGS_STDIO	N/A
DDS_THREAD_SETTINGS_REALTIME_PRIORITY	Set schedule policy to SCHED_FIFO
DDS_THREAD_SETTINGS_PRIORITY_ENFORCE	N/A

6.6 Durable Writer History and Durable Reader State Features

To use the Durable Writer History and Durable Reader State features, you must install a relational database such as MySQL.

In principle, you can use any database that provides an ODBC driver, since ODBC is a standard. However, not all ODBC databases support the same feature set. Therefore, there is no guarantee that the persistent durability features will work with an arbitrary ODBC driver.

We have tested the following driver: MySQL ODBC 5.1.44.

Starting with 4.5e, support for the TimesTen database has been removed.

To use MySQL, you also need MySQL ODBC 5.1.6 (or higher) and UnixODBC 2.2.12 (or higher).

[a]See the Linux programmer's manuals for more information

The Durable Writer History and Durable Reader State features have been tested with the following Linux architectures:

- i86Linux2.6gcc4.1.1
- i86Linux2.6gcc4.6.3
- x64Linux2.6gcc4.1.1
- x64Linux2.6gcc4.6.3

For information on database setup, please see the RTI Connext DDS Core Libraries Getting Started Guide Addendum for Database Setup.

6.7 Libraries Required for Using Distributed Logger

RTI Distributed Logger is supported on all the platforms in Table 6.1 Linux Platforms on ARM CPUs through Table 6.3 Linux Platforms on PowerPC CPUs *except* the following:

- Freescale P2020RDB
- Linux platforms on PowerPC CPUs
- Red Hat Enterprise 4.0, 5.2 with Real-Time Extensions
- Wind River Linux 3 and 4
- Yellow Dog Linux 4.0

To use the Distributed Logger APIs, links against the additional libraries in Table 6.10 Additional Libraries for using RTI Distributed Logger . (Select the files appropriate for your chosen library format.)

Table 6.10 Additional Libraries for using RTI Distributed Logger

Language	Static		Dynamic	
	Release	Debug	Release	Debug
C	librtidlcz.a	librtidlczd.a	librtidlc.so	librtidcd.so
C++ (Traditional API)	librtidlcz.a librtidlcppz.a	librtidlcz.a librtidlcppz.a	librtidlc.so librtidlcpp.so	librtidlcd.so librtidlcppd.so
Java	N/A	N/A	distlog.jar distlogdatamodel.jar	distlogd.jar distlogdatamodeld.jar

6.8 Libraries Required for Using Monitoring

Make sure you are consistent in your use of static, dynamic, debug and release versions of the libraries. For example, if your Connext DDS application is linked with the static release version of the Connext DDS libraries, you will need to also use the static release version of the monitoring library. Do not link both static and dynamic libraries. Similarly, do not mix release and debug libraries.

If you are statically linking your application with DDS libraries and you want to add monitoring to your application, you will also need to statically link the monitoring library. The library cannot be loaded dynamically strictly through the QoS profile because it also depends on DDS to publish its data. Therefore, it depends on DDS; the DDS functionality would cause duplicate symbols to be found resulting, in the termination of the process.

Note: If you plan to use *static* libraries, the RTI library from Table 6.11 Additional Libraries for Using Monitoring must appear *first* in the list of libraries to be linked.

Table 6.11 Additional Libraries for Using Monitoring

Static Release	Static Debug	Dynamic Release	Dynamic Debug
librtimonitoringz.a	librtimonitoringzd.a	librtimonitoring.so	librtimonitoringd.so

6.9 Libraries Required for Using RTI Secure WAN Transport APIs

If you choose to use RTI Secure WAN Transport, it must be downloaded and installed separately. It is only available on specific architectures. See the RTI Secure WAN Transport Release Notes and RTI Secure WAN Transport Installation Guide for details.

To use the Secure WAN Transport APIs, link against the additional libraries in Table 6.12 Additional Libraries for using RTI Secure WAN Transport APIs on UNIX-based Systems. (Select the files appropriate for your chosen library format.)

Table 6.12 Additional Libraries for using RTI Secure WAN Transport APIs on UNIX-based Systems

Library Format	RTI Secure WAN Transport Libraries[a]	OpenSSL Libraries[b]
Dynamic Release	libnddstransportwan.so libnddstransporttls.so	libssl.so libcrypto.so
Dynamic Debug	libnddstransportwand.so libnddstransporttlsd.so	
Static Release	libnddstransporttlsz.a libnddstransporttlszd.a	
Static Debug	libnddstransportwanz.a libnddstransportwanzd.a	

6.10 Libraries Required for Using RTI TCP Transport APIs

To use the TCP Transport APIs, link against the additional libraries inTable 6.13 Additional Libraries for using RTI TCP Transport APIs on UNIX-based Systems . If you are using *RTI TLS Support*, see Table 6.14 Additional Libraries for using RTI TCP Transport APIs on UNIX-based Systems with TLS Enabled. (Select the files appropriate for your chosen library format.)

Table 6.13 Additional Libraries for using RTI TCP Transport APIs on UNIX-based Systems

Library Format	RTI TCP Transport Libraries[c]
Dynamic Release	libnddstransporttcp.so
Dynamic Debug	libnddstransporttcpd.so
Static Release	libnddstransporttcpz.a
Static Debug	libnddstransporttcpzd.a

[a]The libraries are in <NDDSHOME>/lib/<architecture>.

[b]These libraries are in <*openssl install dir*>/<architecture>/lib.

[c]These libraries are in <NDDSHOME>/lib/<architecture>.

Table 6.14 Additional Libraries for using RTI TCP Transport APIs on UNIX-based Systems with TLS Enabled

Library Format	RTI TLS Libraries[a]
Dynamic Release	libnddstls.so
Dynamic Debug	libnddstlsd.so
Static Release	libnddstlsz.a
Static Debug	libnddstlszd.a
OpenSSL Libraries	libssl.so libcrypto.so

[a]These libraries are in <NDDSHOME>/lib/<architecture>.

Chapter 7 LynxOS Platforms

Table 7.1 Supported LynxOS Platforms lists the architectures supported on LynxOS® operating systems.

Table 7.1 Supported LynxOS Platforms

Operating System	CPU	Compiler	RTI Architecture
LynxOS 4.0	x86	gcc 3.2.2	i86Lynx4.0.0gcc3.2.2
	PPC 74xx (such as 7410)	gcc 3.2.2	ppc7400Lynx4.0.0gcc3.2.2
	PPC 604, PPC 7XX (such as 750)	gcc 3.2.2	ppc750Lynx4.0.0gcc3.2.2
LynxOS 4.2	PPC 74xx (such as 7410)	gcc 3.2.2	ppc7400Lynx4.2.0gcc3.2.2
LynxOS 5.0	PPC 74xx (such as 7410)	gcc 3.4.3	ppc7400Lynx5.0.0gcc3.4.3

Table 7.2 Building Instructions for LynxOS Architectures and Table 7.3 Building Instructions for LynxOS Architectures list the compiler flags and libraries you will need to link into your application.

See also:

- Libraries Required for Using Monitoring (Section 7.8 on page 75)

Make sure you are consistent in your use of static, dynamic, debug and release versions of the libraries. Do not link both static and dynamic libraries. Similarly, do not mix release and debug libraries.

Table 7.4 Running Instructions for LynxOS Architectures provides details on the environment variables that must be set at run time for a LynxOS architecture.

Table 7.5 Library-Creation Details for LynxOS Architectures provides details on how the libraries were built by RTI. This table is provided strictly for informational purposes; you do not need to use these parameters to compile your application. You may find this information useful if you are involved in any in-depth debugging.

Note: The Java API is not currently supported on LynxOS platforms. If you would like Java to be supported on LynxOS, please contact your RTI account manager.

Table 7.2 Building Instructions for LynxOS Architectures

API	Library Format[a]	Required RTI Libraries [bcd]
C++ (Traditional and Modern APIs)	Static Release	libnddscppz.a or libnddscpp2z.a libnddscz.a libnddscorez.a librticonnextmsgcppz.a
	Static Debug	libnddscppzd.a or libnddscpp2zd.a libnddsczd.a libnddscorezd.a librticonnextmsgcppzd.a
	Dynamic Release	libnddscpp.so or libnddscpp2.so libnddsc.so libnddscore.so librticonnextmsgcpp.so
	Dynamic Debug	libnddscppd.so or libnddscpp2d.so libnddscd.so libnddscored.so librticonnextmsgcppd.so

[a] Dynamic libraries are not supported under LynxOS-178.

[b] Choose libnddscpp*.* for the Traditional C++ API or libnddscpp2*.* for the Modern C++ API.

[c] The RTI C/C++ libraries are in $(NDDSHOME)/lib/<architecture> (where $(NDDSHOME) is where Connext DDS is installed, see **Paths Mentioned in Documentation (Section 1.1 on page 4)**.

[d] The *rticonnextmsg* library only applies if you have the RTI Connext DDS Professional, Evaluation, or Basic package type. It is not provided with the RTI Connext DDS Core package type.

Table 7.2 Building Instructions for LynxOS Architectures

API	Library Format[a]	Required RTI Libraries [bcd]
C	Static Release	libnddscz.a libnddscorez.a librticonnextmsgcz.a
	Static Debug	libnddsczd.a libnddscorezd.a librticonnextmsgczd.a
	Dynamic Release	libnddsc.so libnddscore.so librticonnextmsgc.so
	Dynamic Debug	libnddscd.so libnddscored.so librticonnextmsgcd.so

Table 7.3 Building Instructions for LynxOS Architectures

API	RTI Architecture	Required System Libraries	Required Compiler Flags
C and C++ (Traditional and Modern APIs)	i86Lynx4.0.0gcc3.2.2	-ldb -lm -lrpc -lc -llynx	-DRTI_LYNX -mthreads -mshared
	ppc7400Lynx4.0.0gcc3.2.2		
	ppc7400Lynx4.2.0gcc3.2.2		
	ppc7400Lynx5.0.0gcc3.4.3		
	ppc750Lynx4.0.0gcc3.2.2		

[a]Dynamic libraries are not supported under LynxOS-178.

[b]Choose libnddscpp*.* for the Traditional C++ API or libnddscpp2*.* for the Modern C++ API.

[c]The RTI C/C++ libraries are in $(NDDSHOME)/lib/<architecture> (where $(NDDSHOME) is where Connext DDS is installed, see **Paths Mentioned in Documentation (Section 1.1 on page 4)**.

[d]The *rticonnextmsg* library only applies if you have the RTI Connext DDS Professional, Evaluation, or Basic package type. It is not provided with the RTI Connext DDS Core package type.

Table 7.4 Running Instructions for LynxOS Architectures

RTI Architecture	Library Format (Release & Debug)	Required Environment Variables
All supported LynxOS architectures	Static	None required
	Dynamic	LD_LIBRARY_PATH= ${NDDSHOME}/lib/<*architecture*>: ${LD_LIBRARY_PATH}

Table 7.5 Library-Creation Details for LynxOS Architectures

RTI Architecture	Library Format (Static & Dynamic)	Compiler Flags Used by RTI
i86Lynx4.0.0gcc3.2.2	Release	-mthreads -mshared -fPIC -D_POSIX_THREADS_CALLS -D__NO_INCLUDE_WARN__ -O -Wall -Wno-unknown-pragmas -DPtrIntType=long -DCPU=I80586 -DTARGET=\"i86Lynx4.0.0gcc3.2.2\" -DNDEBUG -c -Wp,-MD
	Debug	-mthreads -mshared -fPIC -D_POSIX_THREADS_CALLS -D__NO_INCLUDE_WARN__ -g -O -Wall -Wno-unknown-pragmas -DPtrIntType=long -DCPU=I80586 -DTARGET=\"i86Lynx4.0.0gcc3.2.2\" -c -Wp,-MD
ppc7400Lynx4.0.0gcc3.2.2	Release	-mcpu=7400 -maltivec -mabi=altivec -fno-exceptions -mthreads -mshared -fPIC -D_POSIX_THREADS_CALLS -D__NO_INCLUDE_WARN__ -O -Wall -Wno-unknown-pragmas -DPtrIntType=long -DCPU=PPC7400 -DTARGET=\"ppc7400Lynx4.0.0gcc3.2.2\" -DNDEBUG -c -Wp,-MD
	Debug	-mcpu=7400 -maltivec -mabi=altivec -fno-exceptions -mthreads -mshared -fPIC -D_POSIX_THREADS_CALLS -D__NO_INCLUDE_WARN__ -g -O -Wall -Wno-unknown-pragmas -DPtrIntType=long -DCPU=PPC7400 -DTARGET=\"ppc7400Lynx4.0.0gcc3.2.2\" -c -Wp,-MD
ppc7400Lynx4.2.0gcc3.2.2	Release	-mcpu=7400 -maltivec -mabi=altivec -fno-exceptions -mthreads -mshared -fPIC -D_POSIX_THREADS_CALLS -D__NO_INCLUDE_WARN__ -O -Wall -Wno-unknown-pragmas -DPtrIntType=long -DCSREAL_IS_FLOAT -DCPU=PPC7400 -DTARGET=\"ppc7400Lynx4.2.0gcc3.2.2\" -DNDEBUG -c -Wp,-MD
	Debug	-mcpu=7400 -maltivec -mabi=altivec -fno-exceptions -mthreads -mshared -fPIC -D_POSIX_THREADS_CALLS -D__NO_INCLUDE_WARN__ -O -Wall -Wno-unknown-pragmas -DPtrIntType=long -DCSREAL_IS_FLOAT -DCPU=PPC7400 -DTARGET=\"ppc7400Lynx4.2.0gcc3.2.2\" -c -Wp,-MD

Table 7.5 Library-Creation Details for LynxOS Architectures

RTI Architecture	Library Format (Static & Dynamic)	Compiler Flags Used by RTI
ppc7400Lynx5.0.0gcc3.4.3	Release	-mcpu=7400 -maltivec -mabi=altivec -fno-exceptions -mthreads -mshared -fPIC -D_POSIX_THREADS_CALLS -D__NO_INCLUDE_WARN__ -O -Wall -Wno-unknown-pragmas -DPtrIntType=long -DCSREAL_IS_FLOAT -DCPU=PPC7400 -DTARGET=\"ppc7400Lynx5.0.0gcc3.4.3\" -DNDEBUG -c -Wp,-MD
	Debug	-mcpu=7400 -maltivec -mabi=altivec -fno-exceptions -mthreads -mshared -fPIC -D_POSIX_THREADS_CALLS -D__NO_INCLUDE_WARN__ -O -Wall -Wno-unknown-pragmas -DPtrIntType=long -DCSREAL_IS_FLOAT -DCPU=PPC7400 -DTARGET=\"ppc7400Lynx5.0.0gcc3.4.3\" -c -Wp,-MD
ppc750Lynx4.0.0gcc3.2.2	Release	-mcpu=750 -fno-exceptions -mthreads -mshared -fPIC -D_POSIX_THREADS_CALLS -D__NO_INCLUDE_WARN__ -O -Wall -Wno-unknown-pragmas -DPtrIntType=long -DCPU=PPC750 -DTARGET=\"ppc750Lynx4.0.0gcc3.2.2\" -DNDEBUG -c -Wp,-MD
	Debug	-mcpu=750 -fno-exceptions -mthreads -mshared -fPIC -D_POSIX_THREADS_CALLS -D__NO_INCLUDE_WARN__ -g -O -Wall -Wno-unknown-pragmas -DPtrIntType=long -DCPU=PPC750 -DTARGET=\"ppc750Lynx4.0.0gcc3.2.2\" -c -Wp,-MD

7.1 Support for Modern C++ API

Connext DDS provides two different C++ APIs, which we refer to as the "Traditional C++" and "Modern C++" APIs. The Modern C++ API is only available for the LynxOS 5.0 platform.

For more information on Modern C++, see "Traditional vs. Modern C++" in the User's Manual.

7.2 Multicast Support

Multicast is supported on all LynxOS platforms, but it is not configured out of the box. That is, the default value for the initial peers list (NDDS_DISCOVERY_PEERS) does not include a multicast address.

To configure a LynxOS target to use multicast, you need to add routes so multicast packets will be sent via the proper network interfaces. To add routes, use the "route add" command. The specific parameters depend on how the target is configured, the name of the interface (such as **elxl0** in the example below), etc. Please refer to your LynxOS documentation for details on the "route add" command.

For example:

```
route add -net 224.0.0.0 -netmask 240.0.0.0 -interface elxl0
```

Note—Group Address Ignored for Multicast Reception on Loopback: On LynxOS architectures, the multicast-loopback implementation ignores the group address when receiving messages. This causes

Connext DDS to receive all outgoing multicast traffic originating from the host for that port. Thus, if you have two participants on the same host and in the same DDS domain, both listening for discovery traffic over multicast, they will discover each other, regardless of the multicast address to which they are listening. (The correct behavior would be to receive messages only for the addresses to which the current process (not the host) is subscribed.)

7.3 Supported Transports

Shared memory: Supported and enabled by default.

UDPv4: Supported and enabled by default.

UDPv6: Not supported.

TCP/IPv4: Not supported.

7.3.1 Shared Memory Support

To see a list of shared memory resources in use, use the '**ipcs**' command. To clean up shared memory and shared semaphore resources, use the '**ipcrm**' command.

The shared memory keys used by Connext DDS are in the range of 0x400000. For example:

```
ipcs -m | grep 0x004
```

The shared semaphore keys used by Connext DDS are in the range of 0x800000; the shared mutex keys are in the range of 0xb00000. For example:

```
ipcs -s | grep 0x008

ipcs -s | grep 0x00b
```

Please refer to the shared-memory transport online documentation for details on the shared memory and semaphore keys used by Connext DDS.

7.4 Monotonic Clock Support

The monotonic clock (described in "Clock Selection" in the User's Manual) is not supported on LynxOS platforms.

7.5 Thread Configuration

Table 7.6 Thread Settings for LynxOS Platforms lists the thread settings for LynxOS platforms.

Table 7.7 Thread-Priority Definitions for LynxOS Platforms lists the thread-priority definitions.

7.5.1 Support for Controlling CPU Core Affinity for RTI Threads

Support for controlling CPU core affinity (described in "Controlling CPU Core Affinity" in the User's Manual) is not available for LynxOS platforms.

Table 7.6 Thread Settings for LynxOS Platforms

Applicable Thread	DDS_ ThreadSettings_t	Platform-Specific Setting
Asynchronous Publisher, Asynchronous flushing thread	mask	OS default thread type
	priority	17
	stack_size	16*1024
	cpu_list	CPU core affinity not supported
	cpu_rotation	CPU core affinity not supported
Database thread	mask	DDS_THREAD_SETTINGS_STDIO
	priority	10
	stack_size	16*1024
	cpu_list	CPU core affinity not supported
	cpu_rotation	CPU core affinity not supported
Event thread	mask	DDS_THREAD_SETTINGS_STDIO | DDS_THREAD_SETTINGS_ FLOATING_POINT
	priority	13
	stack_size	4*16*1024
	cpu_list	CPU core affinity not supported
	cpu_rotation	CPU core affinity not supported
ReceiverPool threads	mask	DDS_THREAD_SETTINGS_STDIO | DDS_THREAD_SETTINGS_ FLOATING_POINT
	priority	29
	stack_size	4*16*1024
	cpu_list	CPU core affinity not supported
	cpu_rotation	CPU core affinity not supported

Table 7.7 Thread-Priority Definitions for LynxOS Platforms

Thread-Priority Definition	Operating-System Priority
THREAD_PRIORITY_DEFAULT	17
THREAD_PRIORITY_HIGH	32
THREAD_PRIORITY_ABOVE_NORMAL	29
THREAD_PRIORITY_NORMAL	17
THREAD_PRIORITY_BELOW_NORMAL	13
THREAD_PRIORITY_LOW	10

7.6 Durable Writer History and Durable Reader State Features

The Durable Writer History and Durable Reader State features are not supported on LynxOS platforms.

7.7 Distributed Logger Support

RTI Distributed Logger is not supported on LynxOS platforms.

7.8 Libraries Required for Using Monitoring

Make sure you are consistent in your use of static, dynamic, debug and release versions of the libraries. For example, if your Connext DDS application is linked with the static release version of the Connext DDS libraries, you will need to also use the static release version of the monitoring library. Do not link both static and dynamic libraries. Similarly, do not mix release and debug libraries.

If you are statically linking your application with DDS libraries and you want to add monitoring to your application, you will also need to statically link the monitoring library. The library cannot be loaded dynamically strictly through the QoS profile because it also depends on DDS to publish its data. Therefore, it depends on DDS; the DDS functionality would cause duplicate symbols to be found resulting, in the termination of the process.

Note: If you are plan to use *static* libraries, the RTI library from Table 7.8 Additional Libraries for Using Monitoring must appear *first* in the list of libraries to be linked.

Table 7.8 Additional Libraries for Using Monitoring

Static Release	Static Debug	Dynamic Release	Dynamic Debug
librtimonitoringz.a	librtimonitoringzd.a	librtimonitoring.so	librtimonitoringd.so

7.9 IP Fragmentation Issues

The LynxOS platforms do not support IP fragmentation over the loopback interface due to a bug in the OS (see below). The maximum size of a UDP packet that can be sent over the loopback interface is therefore limited by the size of the MTU on this interface, which by default is 16384 bytes. Since the default **message_size_max** for the builtin-UDPv4 transport is 65507 bytes (the maximum UDP user payload), you must adjust the size of the MTU of the loopback interface to accommodate UDP messages larger than 16384 bytes (including the UDP header). You can increase the size of the MTU with the following command:

```
> ifconfig lo0 mtu 65535
```

Note: The maximum size of the MTU on the loopback interface is 65535, which will allow RTPS payloads of 65507 bytes.

For more information on this issue, contact LynuxWorks Support about bug #30191.

Chapter 8 OS X Platforms

Table 8.1 Supported OS X Platforms lists the architectures supported on Mac OS X operating systems.

Table 8.1 Supported OS X Platforms

Operating System	CPU	Compiler	RTI Architecture Abbreviation
OS X 10.8	x64	clang 4.1 Java Platform, Standard Edition JDK 1.7 or 1.8	x64Darwin12clang4.1
OS X 10.10	x64	clang 6.0 Java Platform, Standard Edition JDK 1.7 or 1.8	x64Darwin14clang6.0
OS X 10.11 (see Notes below)	x64	clang 7.0 Java Platform, Standard Edition JDK 1.7 or 1.8	x64Darwin15clang7.0

Table 8.2 Building Instructions for OS X Architectures lists the compiler flags and libraries you will need to link into your application.

See also:

- Libraries Required for Using Distributed Logger (Section 8.7 on page 85)
- Libraries Required for Using Monitoring (Section 8.8 on page 85)
- Libraries Required for Using RTI Secure WAN Transport APIs (Section 8.9 on page 86)
- Libraries Required for Using RTI TCP Transport APIs (Section 8.10 on page 87)

Make sure you are consistent in your use of static, dynamic, debug and release versions of the libraries. Do not link both static and dynamic libraries. Similarly, do not mix release and debug libraries.

Table 8.3 Running Instructions for OS X Architectures provides details on the environment variables that must be set at run time for an OS X architecture.

Table 8.4 Library-Creation Details for OS X Architectures provides details on how the libraries were built by RTI. This table is provided strictly for informational purposes; you do not need to use these parameters to compile your application. You may find this information useful if you are involved in any in-depth debugging.

Notes for OS X 10.11:

- Due to the SIP security feature introduced in OS X 10.11, the **runPub/Sub.sh** scripts for the Java shipped examples will not work if the java binary is under one of the folders affected by the feature (i.e. **/usr/bin**/). To make them run, the java binary must be outside of the affected folders.
- The System Integrity Protection feature introduced in OS X 10.11 makes it impossible for the scripts under **<NDDSHOME>/bin** to pick up the value of the DYLD_LIBRARY_PATH environment variable at run time. To workaround this issue, Connext DDS 5.2.3 introduces RTI_LD_LIBRARY_PATH, an alternative environment variable that can be used in lieu of DYLD_LIBRARY_PATH and LD_LIBRARY_PATH to add library paths on UNIX-like systems.

  For example, to add **<OPENSSLHOME>/lib** and **<NDDSHOME/lib/<architecture>** (i.e., the library paths required for running RTI Routing Service with the Secure WAN or TLS transports), export the RTI_LD_LIBRARY_PATH environment variable and run Routing Service as follows:

```
cd <NDDSHOME>
export RTI_LD_LIBRARY_PATH=<OPENSSLHOME>/lib:<NDDSHOME>/lib/<ARCHITECTURE>
./bin/rtiroutingservice -cfgName <your_configuration>
```

Table 8.2 Building Instructions for OS X Architectures

API	Library Format	Required RTI Libraries[abc]	Required System Libraries	Required Compiler Flags
C++ (Traditional and Modern APIs)	Static Release	libnddscppz.a or libnddscpp2z.a libnddscz.a libnddscorez.a librticonnextmsgcppz.a	-ldl -lm -lpthread	-dynamic -lpthread -lc -single_module -DRTI_UNIX -DRTI_DARWIN -DRTI_DARWIN10 -DRTI_64BIT
	Static Debug	libnddscppzd.a or libnddscpp2zd.a libnddsczd.a libnddscorezd.a librticonnextmsgcppzd.a		
	Dynamic Release	libnddscpp.dylib or libnddscpp2.dylib libnddsc.dylib libnddscore.dylib librticonnextmsgcpp.dylib		
	Dynamic Debug	libnddscppd.dylib or libnddscpp2d.dylib libnddscd.dylib libnddscored.dylib librticonnextmsgcppd.dylib		

[a]Choose libnddscpp*.* for the Traditional C++ API or libnddscpp2*.* for the Modern C++ API.

[b]The Connext DDS C/C++ libraries are in <NDDSHOME>/lib/<architecture>/. <NDDSHOME> is where Connext DDS is installed, see Paths Mentioned in Documentation (Section 1.1 on page 4)

[c]The ***rticonnextmsg*** library only applies if you have the RTI Connext DDS Professional, Evaluation, or Basic package type. It is not provided with the RTI Connext DDS Core package type.

Table 8.2 Building Instructions for OS X Architectures

API	Library Format	Required RTI Libraries[abc]	Required System Libraries	Required Compiler Flags
C	Static Release	libnddscz.a libnddscorez.a librticonnextmsgcz.a	-ldl -lm -lpthread	-dynamic -lpthread -lc -single_module -DRTI_UNIX -DRTI_DARWIN -DRTI_DARWIN10 -DRTI_64BIT
	Static Debug	libnddsczd.a libnddscorezd.a librticonnextmsgczd.a		
	Dynamic Release	libnddsc.dylib libnddscore.dylib librticonnextmsgc.dylib		
	Dynamic Debug	libnddscd.dylib libnddscored.dylib librticonnextmsgcd.dylib		
Java	Release	nddsjava.jar rticonnextmsg.jar	N/A	None required
	Debug	nddsjavad.jar rticonnextmsgd.jar		

[a]Choose libnddscpp*.* for the Traditional C++ API or libnddscpp2*.* for the Modern C++ API.

[b]The Connext DDS C/C++ libraries are in <NDDSHOME>/lib/<architecture>/. <NDDSHOME> is where Connext DDS is installed, see Paths Mentioned in Documentation (Section 1.1 on page 4)

[c]The ***rticonnextmsg*** library only applies if you have the RTI Connext DDS Professional, Evaluation, or Basic package type. It is not provided with the RTI Connext DDS Core package type.

Table 8.3 Running Instructions for OS X Architectures

RTI Architecture	Library Format (Release & Debug)	Required Environment Variables[a]
x64Darwin12clang4.1	Static	None required
	Dynamic	DYLD_LIBRARY_PATH=${NDDSHOME}/lib/x64Darwin12clang4.1:${DYLD_LIBRARY_PATH}
x64Darwin12clang4.1 for Java	N/A	DYLD_LIBRARY_PATH=${NDDSHOME}/lib/x64Darwin12clang4.1:${DYLD_LIBRARY_PATH}
x64Darwin14clang6.0	Static	None required
	Dynamic	DYLD_LIBRARY_PATH=${NDDSHOME}/lib/x64Darwin14clang6.0:${DYLD_LIBRARY_PATH}
x64Darwin14clang6.0 for Java	N/A	DYLD_LIBRARY_PATH=${NDDSHOME}/lib/x64Darwin14clang6.0:${DYLD_LIBRARY_PATH}
x64Darwin15clang7.0	Static	None required
	Dynamic	DYLD_LIBRARY_PATH=${NDDSHOME}/lib/x64Darwin15clang7.0:${DYLD_LIBRARY_PATH}
x64Darwin15clang7.0 for Java	N/A	DYLD_LIBRARY_PATH=${NDDSHOME}/lib/x64Darwin15clang7.0:${DYLD_LIBRARY_PATH}

[a] ${NDDSHOME} is where Connext DDS is installed. ${DYLD_LIBRARY_PATH} represents the value of the DYLD_LIBRARY_PATH variable prior to changing it to support Connext DDS. When using nddsjava.jar, the Java virtual machine (JVM) will attempt to load release versions of the native libraries (nddsjava.dylib, nddscore.dylib, nddsc.dylib). When using nddsjavad.jar, the JVM will attempt to load debug versions of the native libraries (nddsjava.dylib, nddscore.dylib, nddsc.dylib).

Table 8.4 Library-Creation Details for OS X Architectures

RTI Architecture	Library Format (Static & Dynamic)	Compiler Flags Used by RTI
x64Darwin12clang4.1	Release	-O -Wall -Wno-unknown-pragmas -DPtrIntType=long -DCSREAL_IS_FLOAT -DCPU=AMD64 -DTARGET=\"x64Darwin12clang4.1\" -c -Wp,-MD
	Debug	-g -Wall -Wno-unknown-pragmas -DPtrIntType=long -DCSREAL_IS_FLOAT -DCPU=AMD64 -DTARGET=\" x64Darwin12clang4.1\" -c -Wp,-MD
x64Darwin12clang4.1 for Java	Release	-target 1.4 -source 1.4
	Debug	-target 1.4 -source 1.4 -g
x64Darwin14clang6.0	Release	-O -Wall -Wno-unknown-pragmas -DPtrIntType=long -DCSREAL_IS_FLOAT -DCPU=AMD64 -DTARGET=\"x64Darwin14gcc6.0\" -c -Wp,-MD
	Debug	-g -Wall -Wno-unknown-pragmas -DPtrIntType=long -DCSREAL_IS_FLOAT -DCPU=AMD64 -DTARGET=\" x64Darwin14gcc6.0\" -c -Wp,-MD
x64Darwin14clang6.0 for Java	Release	-target 1.4 -source 1.4
	Debug	-target 1.4 -source 1.4 -g
x64Darwin15clang7.0	Release	/usr/bin/clang++ -x c -arch x86_64 -mtune=core2 -Wno-trigraphs -fpascal-strings -fasm-blocks -O -Wall -Wno-unknown-pragmas -DPtrIntType=long -DCSREAL_IS_FLOAT -DCPU=AMD64 -DNDEBUG
	Debug	/usr/bin/clang++ -x c -arch x86_64 -mtune=core2 -Wno-trigraphs -fpascal-strings -fasm-blocks -Wall -Wno-unknown-pragmas -DPtrIntType=long -DCSREAL_IS_FLOAT -DCPU=AMD64
x64Darwin15clang7.0 for Java	Release	-target 1.4 -source 1.4
	Debug	-target 1.4 -source 1.4 -g

8.1 Support for Modern C++ API

Connext DDS provides two different C++ APIs, which we refer to as the "Traditional C++" and "Modern C++" APIs. The Modern C++ API is available for all OS X platforms.

For more information on Modern C++, see "Traditional vs. Modern C++" in the User's Manual.

8.2 Multicast Support

Multicast is supported on OS X platforms and is configured out of the box. That is, the default value for the initial peers list (**NDDS_DISCOVERY_PEERS**) includes a multicast address. See the online documentation for more information.

8.3 Supported Transports

Shared memory: Supported and enabled by default.

UDPv4: Supported and enabled by default.

UDPv6: Not supported.

TCP/IPv4: Supported.

8.4 Monotonic Clock Support

The monotonic clock (described in "Clock Selection" in the User's Manual) is not supported on OS X platforms.

8.5 Thread Configuration

Table 8.5 Thread Settings for OS X Platforms lists the thread settings for OS X platforms.

Table 8.6 Thread-Priority Definitions for OS X Platforms lists the thread-priority definitions.

8.5.1 Support for Controlling CPU Core Affinity for RTI Threads

Support for controlling CPU core affinity (described in "Controlling CPU Core Affinity" in the User's Manual) is not available for OS X platforms.

Table 8.5 Thread Settings for OS X Platforms

Applicable Thread	DDS_ ThreadSettings_ t	Platform-Specific Setting
Asynchronous Publisher, Asynchronous flushing thread	mask	OS default thread type
	priority	OS default thread priority
	stack_size	OS default thread stack size
	cpu_list	CPU core affinity not supported
	cpu_rotation	CPU core affinity not supported
Database thread	mask	DDS_THREAD_SETTINGS_STDIO
	priority	OS default thread priority
	stack_size	OS default thread stack size
	cpu_list	CPU core affinity not supported
	cpu_rotation	CPU core affinity not supported
Event thread	mask	DDS_THREAD_SETTINGS_STDIO | DDS_THREAD_ SETTINGS_FLOATING_POINT
	priority	OS default thread priority
	stack_size	OS default thread stack size
	cpu_list	CPU core affinity not supported
	cpu_rotation	CPU core affinity not supported
ReceiverPool threads	mask	DDS_THREAD_SETTINGS_STDIO | DDS_THREAD_ SETTINGS_FLOATING_POINT
	priority	OS default thread priority
	stack_size	OS default thread stack size
	cpu_list	CPU core affinity not supported
	cpu_rotation	CPU core affinity not supported

Table 8.6 Thread-Priority Definitions for OS X Platforms

Thread-Priority Definition	Operating-System Priority
THREAD_PRIORITY_DEFAULT	If any of these constants are used to set the priority of the thread in the QoS, the OS's default thread priority will be used.
THREAD_PRIORITY_HIGH	
THREAD_PRIORITY_ABOVE_NORMAL	
THREAD_PRIORITY_NORMAL	
THREAD_PRIORITY_BELOW_NORMAL	
THREAD_PRIORITY_LOW	

8.6 Durable Writer History and Durable Reader State Features

The Durable Writer History and Durable Reader State features are not supported on OS X platforms.

8.7 Libraries Required for Using Distributed Logger

RTI Distributed Logger is supported on OS X platforms. Table 8.7 Additional Libraries for using RTI Distributed Logger lists the additional libraries you will need in order to use *Distributed Logger*.

Table 8.7 Additional Libraries for using RTI Distributed Logger

Language	Static		Dynamic	
	Release	Debug	Release	Debug
C++ (Traditional API)	librtidlcz.a librtidlcppz.a	librtidlczd.a librtidlcppzd.a	librtidlc.dylib librtidlcpp.dylib	librtidlcd.dylib librtidlcppd.dylib
C	librtidlcz.a	librtidlczd.a	librtidlc.dylib	librtidlcd.dylib
Java	N/A	N/A	distlog.jar distlogdatamodel.jar	distlogd.jar distlogdatamodeld.jar

8.8 Libraries Required for Using Monitoring

Make sure you are consistent in your use of static, dynamic, debug and release versions of the libraries. For example, if your Connext DDS application is linked with the static release version of the Connext DDS libraries, you will need to also use the static release version of the monitoring library. Do not link both static and dynamic libraries. Similarly, do not mix release and debug libraries.

If you are statically linking your application with DDS libraries and you want to add monitoring to your application, you will also need to statically link the monitoring library. The library cannot be loaded dynamically strictly through the QoS profile because it also depends on DDS to publish its data. Therefore, it depends on DDS; the DDS functionality would cause duplicate symbols to be found resulting, in the termination of the process.

Note: If you are plan to use *static* libraries, the RTI library from Table 8.8 Additional Libraries for Using Monitoring must appear *first* in the list of libraries to be linked.

Table 8.8 Additional Libraries for Using Monitoring

Static Release	Static Debug	Dynamic Release	Dynamic Debug
librtimonitoringz.a	librtimonitoringzd.a	librtimonitoring.dylib	librtimonitoringd.dylib

8.9 Libraries Required for Using RTI Secure WAN Transport APIs

If you choose to use *RTI Secure WAN Transport*, it must be downloaded and installed separately. It is available on all Mac OS X architectures. See the RTI Secure WAN Transport Release Notes and RTI Secure WAN Transport Installation Guide for details.

To use the Secure WAN Transport APIs, link against the additional libraries in Table 8.9 Additional Libraries for using RTI Secure WAN Transport APIs on OS X Systems. (Select the files appropriate for your chosen library format.)

Table 8.9 Additional Libraries for using RTI Secure WAN Transport APIs on OS X Systems

Library Format	RTI Secure WAN Transport Libraries[a]	OpenSSL Libraries[b]
Dynamic Release	libnddstransportwan.dylib libnddstransporttls.dylib	libssl.so libcrypto.so
Dynamic Debug	libnddstransportwand.dylib libnddstransporttlsd.dylib	
Static Release	libnddstransporttlsz.a libnddstransporttlszd.a	
Static Debug	libnddstransportwanz.a libnddstransportwanzd.a	

[a]The libraries are in <NDDSHOME>/lib/<architecture>.

[b]These libraries are in <*openssl install dir*>/<architecture>/lib.

8.10 Libraries Required for Using RTI TCP Transport APIs

To use the TCP Transport APIs, link against the additional libraries in Table 8.10 Additional Libraries for using RTI TCP Transport APIs on OS X Systems . If you are using *RTI TLS Support*, see Table 8.11 Additional Libraries for using RTI TCP Transport APIs on OS X Systems with TLS Enabled. (Select the files appropriate for your chosen library format.)

Table 8.10 Additional Libraries for using RTI TCP Transport APIs on OS X Systems

Library Format	RTI TCP Transport Libraries[a]
Dynamic Release	libnddstransporttcp.so
Dynamic Debug	libnddstransporttcpd.so
Static Release	libnddstransporttcpz.a
Static Debug	libnddstransporttcpzd.a

Table 8.11 Additional Libraries for using RTI TCP Transport APIs on OS X Systems with TLS Enabled

Library Format	RTI TLS Libraries[b]
Dynamic Release	libnddstls.so
Dynamic Debug	libnddstlsd.so
Static Release	libnddstlsz.a
Static Debug	libnddstlszd.a
OpenSSL Libraries	libssl.so libcrypto.so

[a]These libraries are in <NDDSHOME>/lib/<architecture>.

[b]These libraries are in <NDDSHOME>/lib/<architecture>.

Chapter 9 QNX Platforms

Table 9.1 Supported QNX Platforms lists the architectures supported on QNX operating systems.

Table 9.1 Supported QNX Platforms[a]

Operating System	CPU	Compiler	RTI Architecture
QNX Neutrino 6.4.1	x86	qcc 4.3.3 with GNU C++ libraries	i86QNX6.4.1qcc_gpp
QNX Neutrino 6.5	x86	qcc 4.4.2 with GNU C++ libraries	i86QNX6.5qcc_gpp4.4.2
QNX Neutrino 6.5.0 SP1	ARMv7a Cortex	qcc 4.4.2 with Dinkum libraries	armv7aQNX6.5.0SP1qcc_cpp4.4.2

Table 9.2 Building Instructions for QNX Architectures lists the libraries you will need to link into your application.

See also:

- Libraries Required for Using Distributed Logger (Section 9.8 on page 95)
- Libraries Required for Using RTI Secure WAN Transport APIs (Section 9.10 on page 96)
- Libraries Required for Using RTI TCP Transport APIs (Section 9.11 on page 96)

Make sure you are consistent in your use of static, dynamic, debug and release versions of the libraries. Do not link both static and dynamic libraries. Similarly, do not mix release and debug libraries.

Table 9.3 Running Instructions for QNX Architectures provides details on the environment variables that must be set at run time for a QNX architecture.

---

[a]For use with Windows, Linux or Solaris Host as supported by QNX & RTI

Table 9.4 Library-Creation Details for QNX Architectures provides details on how the QNX libraries were built.

Table 9.2 Building Instructions for QNX Architectures

API	Library Format	RTI Libraries[abc]	Required System Libraries	Required Compiler Flags
C++ (Traditional and Modern APIs)	Static Release	libnddscppz.a or libnddscpp2z.a libnddscz.a libnddscorez.a librticonnextmsgcppz.a	-lm -lsocket	-DRTI_QNX
	Static Debug	libnddscppzd.a or libnddscpp2zd.a libnddsczd.a libnddscorezd.a librticonnextmsgcppzd.a		
	Dynamic Release	libnddscpp.so or libnddscpp2.so libnddsc.so libnddscore.so librticonnextmsgcpp.so		
	Dynamic Debug	libnddscppd.so or libnddscpp2d.so libnddscd.so libnddscored.so librticonnextmsgcppd.so		

[a]Choose libnddscpp*.* for the Traditional C++ API or libnddscpp2*.* for the Modern C++ API.

[b]The DDS C/C++ libraries are in $(NDDSHOME)/lib/<***architecture***>.

[c]The ***rticonnextmsg*** library only applies if you have the RTI Connext DDS Professional, Evaluation, or Basic package type. It is not provided with the RTI Connext DDS Core package type.

Table 9.2 Building Instructions for QNX Architectures

API	Library Format	RTI Libraries[abc]	Required System Libraries	Required Compiler Flags
C	Static Release	libnddscz.a libnddscorez.a librticonnextmsgcz.a	-lm -lsocket	-DRTI_QNX
	Static Debug	libnddsczd.a libnddscorezd.a librticonnextmsgczd.a		
	Dynamic Release	libnddsc.so libnddscore.so librticonnextmsgc.so		
	Dynamic Debug	libnddscd.so libnddscored.so librticonnextmsgcd.so		

Table 9.3 Running Instructions for QNX Architectures

RTI Architecture	Library Format (Release & Debug)	Environment Variables
All supported QNX architectures	Static	None required
	Dynamic	LD_LIBRARY_PATH= ${NDDSHOME}/lib/<architecture>: ${LD_LIBRARY_PATH}[d]

[a]Choose libnddscpp*.* for the Traditional C++ API or libnddscpp2*.* for the Modern C++ API.

[b]The DDS C/C++ libraries are in $(NDDSHOME)/lib/<***architecture***>.

[c]The ***rticonnextmsg*** library only applies if you have the RTI Connext DDS Professional, Evaluation, or Basic package type. It is not provided with the RTI Connext DDS Core package type.

[d]${NDDSHOME} represents the root directory of your Connext DDS installation. ${LD_LIBRARY_PATH} represents the value of the LD_LIBRARY_PATH variable prior to changing it to support Connext DDS. When using nddsjava.jar, the Java virtual machine (JVM) will attempt to load release versions of the native libraries. When using nddsjavad.jar, the JVM will attempt to load debug versions of the native libraries.

Table 9.4 Library-Creation Details for QNX Architectures

RTI Architecture	Library Format (Static & Dynamic)	Compiler Flags Used by RTI
armv7aQNX6.5.0SP1qcc_cpp4.4.2	Release	qcc -Vgcc/4.4.2,gcc_ntoarmv7le_cpp -fPIC -fexceptions -DFD_SETSIZE=512 -O -Wall -Wno-unknown-pragmas -DRTS_QNX -DPtrIntType=long -DCSREAL_IS_FLOAT -DCPU=ARMV7 -DTARGET=\"armv7aQNX6.5.0SP1qcc_cpp4.4.2\" -DNDEBUG
	Debug	qcc -Vgcc/4.4.2,gcc_ntoarmv7le_cpp -fPIC -fexceptions -DFD_SETSIZE=512 -g -Wall -Wno-unknown-pragmas -DRTS_QNX -DPtrIntType=long -DCSREAL_IS_FLOAT -DCPU=ARMV7 -DTARGET=\"armv7aQNX6.5.0SP1qcc_cpp4.4.2\"
i86QNX6.4.1qcc_gpp	Release	qcc -Vgcc/4.3.3,gcc_ntox86 -Y_gpp -lang-c -fPIC -fexceptions -O -Wall -Wno-unknown-pragmas -DNDEBUG
	Debug	qcc -Vgcc/4.3.3,gcc_ntox86 -Y_gpp -lang-c -fPIC -fexceptions -g -Wall -Wno-unknown-pragmas
i86QNX6.5qcc_gpp4.4.2	Release	qcc -Vgcc/4.4.2,gcc_ntox86 -Y_gpp -m32 -march=i386 -mtune=generic -fPIC -fexceptions -DFD_SETSIZE=512 -O -Wall -Wno-unknown-pragmas -DRTS_QNX -DPtrIntType=long -DCSREAL_IS_FLOAT -DCPU=I80586 -DTARGET=\"i86QNX6.5qcc_gpp4.4.2\" -DNDEBUG
	Debug	qcc -Vgcc/4.4.2,gcc_ntox86 -Y_gpp -m32 -march=i386 -mtune=generic -fPIC -fexceptions -DFD_SETSIZE=512 -g -Wall -Wno-unknown-pragmas -DRTS_QNX-DPtrIntType=long -DCSREAL_IS_FLOAT -DCPU=I80586 -DTARGET=\"i86QNX6.5qcc_gpp4.4.2\"

9.1 Required Change for Building with C++ Libraries for QNX Platforms

For QNX architectures in Connext DDS 5.0 and higher:

The C++ libraries are now built *without* the **-fno-rtti** flag and *with* the **-fexceptions** flag. To build QNX architectures with Connext DDS 5.0 and higher, you must build your C++ applications *without* **-fno-exceptions** in order to link with the RTI libraries. In summary:

- Do *not* use **-fno-exceptions** when building a C++ application or the build will fail. It is not necessary to use -fexceptions, but doing so will not cause a problem.
- It is no longer necessary to use **-fno-rtti**, but doing so will not cause a problem.

9.2 Support for Modern C++ API

Connext DDS provides two different C++ APIs, which we refer to as the "Traditional C++" and "Modern C++" APIs. The Modern C++ API is available for all QNX platforms.

For more information on Modern C++, see "Traditional vs. Modern C++" in the User's Manual.

9.3 Multicast Support

Multicast is supported on QNX platforms and is configured out of the box. That is, the default value for the initial peers list (NDDS_DISCOVERY_PEERS) includes a multicast address. See the online documentation for more information.

9.4 Supported Transports

Shared Memory: Supported and enabled by default.

To see a list of the shared memory resources, enter:

```
'ls /dev/shmem/RTIOsapiSharedMemorySegment-*'
```

To clean up the shared memory resources, remove the files listed in **dev/shmem/**. The shared resource names used by Connext DDS begin with **'RTIOsapiSharedMemorySem-'**. To see a list of shared semaphores, enter:

```
'ls /dev/sem/RTIOsapiSharedMemorySemMutex*'
```

To clean up the shared semaphore resources, remove the files listed in **/dev/sem/'**.

The permissions for the semaphores created by Connext DDS are modified by the process' **umask** value. If you want to have shared memory support between different users, run the command "**umask 000**" to change the default **umask** value to 0 before running your Connext DDS application.

UDPv4: Supported and enabled by default.

UDPv6: Supported. The transport is not enabled by default; the peers list must be modified to support IPv6. No Traffic Class support.

To use the UDPv6 transport, the network stack must provide IPv6 capability. Enabling UDPv6 may involve switching the network stack server and setting up IPv6 route entries.

TCP/IPv4: Supported on i86QNX6.5qcc_cpp4.4.2 and armv7aQNX6.5.0SP1qcc_cpp4.4.2.

TLS: Supported on i86QNX6.5qcc_cpp4.4.2 and armv7aQNX6.5.0SP1qcc_cpp4.4.2.

9.5 Monotonic Clock Support

The monotonic clock (described in "Clock Selection" in the User's Manual) is supported on QNX platforms.

9.6 Thread Configuration

Table 9.5 Thread Settings for QNX Platforms lists the thread settings for QNX platforms.

Table 9.6 Thread-Priority Definitions for QNX Platforms lists the thread-priority definitions.

9.6.1 Support for Controlling CPU Core Affinity for RTI Threads

Support for controlling CPU core affinity (described in "Controlling CPU Core Affinity" in the User's Manual) is not available for QNX platforms.

Table 9.5 Thread Settings for QNX Platforms

Applicable Thread	DDS_ ThreadSettings_t	Platform-Specific Setting
Asynchronous Publisher, Asynchronous flushing thread	mask	OS default thread type
	priority	10
	stack_size	32 * 1024
	cpu_list	CPU core affinity not supported
	cpu_rotation	CPU core affinity not supported
Database thread	mask	DDS_THREAD_SETTINGS_STDIO
	priority	8
	stack_size	32 * 1024
	cpu_list	CPU core affinity not supported
	cpu_rotation	CPU core affinity not supported

Table 9.5 Thread Settings for QNX Platforms

Applicable Thread	DDS_ ThreadSettings_t	Platform-Specific Setting
Event thread	mask	DDS_THREAD_SETTINGS_STDIO | DDS_THREAD_SETTINGS_ FLOATING_POINT
	priority	9
	stack_size	4 * 32 * 1024
	cpu_list	CPU core affinity not supported
	cpu_rotation	CPU core affinity not supported
ReceiverPool threads	mask	DDS_THREAD_SETTINGS_STDIO | DDS_THREAD_SETTINGS_ FLOATING_POINT
	priority	40
	stack_size	4 * 32 * 1024
	cpu_list	CPU core affinity not supported
	cpu_rotation	CPU core affinity not supported

Table 9.6 Thread-Priority Definitions for QNX Platforms

Thread-Priority Definition	Operating-System Priority
THREAD_PRIORITY_DEFAULT	10
THREAD_PRIORITY_HIGH	60
THREAD_PRIORITY_ABOVE_NORMAL	40
THREAD_PRIORITY_NORMAL	10
THREAD_PRIORITY_BELOW_NORMAL	9
THREAD_PRIORITY_LOW	8

9.7 Durable Writer History and Durable Reader State Features

The Durable Writer History and Durable Reader State features are not supported on QNX platforms.

9.8 Libraries Required for Using Distributed Logger

RTI Distributed Logger is supported on QNX platforms on x86 CPUs. It is not supported on QNX platforms on ARM CPUs.

Table 9.7 Additional Libraries for using RTI Distributed Logger lists the additional libraries you will need in order to use *Distributed Logger*.

Table 9.7 Additional Libraries for using RTI Distributed Logger

Language	Static		Dynamic	
	Release	Debug	Release	Debug
C	librtidlcz.a	librtidlczd.a	librtidlc.so	librtidcd.so
C++ (Traditional API)	librtidlcz.a librtidlcppz.a	librtidlczd.a librtidlcppzd.a	librtidlc.so librtidlcpp.so	librtidlcd.so librtidlcppd.so

9.9 Libraries Required for Using Monitoring

Make sure you are consistent in your use of static, dynamic, debug and release versions of the libraries. For example, if your Connext DDS application is linked with the static release version of the Connext DDS libraries, you will need to also use the static release version of the monitoring library. Do not link both static and dynamic libraries. Similarly, do not mix release and debug libraries.

If you are statically linking your application with DDS libraries and you want to add monitoring to your application, you will also need to statically link the monitoring library. The library cannot be loaded dynamically strictly through the QoS profile because it also depends on DDS to publish its data. Therefore, it depends on DDS; the DDS functionality would cause duplicate symbols to be found resulting, in the termination of the process.

Note: If you are plan to use *static* libraries, the RTI library from Table 9.8 Additional Libraries for Using Monitoring must appear *first* in the list of libraries to be linked.

Table 9.8 Additional Libraries for Using Monitoring

Static Release	Static Debug	Dynamic Release	Dynamic Debug
librtimonitoringz.a	librtimonitoringzd.a	librtimonitoring.so[a]	librtimonitoringd.so[b]

[a] To use dynamic libraries, make sure the permissions on the .so library files are readable by everyone.

[b] To use dynamic libraries, make sure the permissions on the .so library files are readable by everyone.

9.10 Libraries Required for Using RTI Secure WAN Transport APIs

If you choose to use *RTI Secure WAN Transport*, it must be downloaded and installed separately. It is only available for QNX 6.5 architectures. See the RTI Secure WAN Transport Release Notes and RTI Secure WAN Transport Installation Guide for details.

To use the Secure WAN Transport APIs, link against the additional libraries in Table 9.9 Additional Libraries for using RTI Secure WAN Transport APIs on QNX 6.5 Systems. (Select the files appropriate for your chosen library format.)

Table 9.9 Additional Libraries for using RTI Secure WAN Transport APIs on QNX 6.5 Systems

Library Format	RTI Secure WAN Transport Libraries[a]	OpenSSL Libraries[b]
Dynamic Release	libnddstransportwan.so libnddstransporttls.so	libssl.so libcrypto.so
Dynamic Debug	libnddstransportwand.so libnddstransporttlsd.so	
Static Release	libnddstransporttlsz.a libnddstransporttlszd.a	
Static Debug	libnddstransportwanz.a libnddstransportwanzd.a	

9.11 Libraries Required for Using RTI TCP Transport APIs

To use the TCP Transport APIs, link against the additional libraries in Table 9.10 Additional Libraries for using RTI TCP Transport APIs on QNX 6.5 Systems . It is only available for QNX 6.5 architectures. If you are using *RTI TLS Support*, see Table 9.11 Additional Libraries for using RTI TCP Transport APIs on QNX Systems with TLS Enabled. (Select the files appropriate for your chosen library format.)

Table 9.10 Additional Libraries for using RTI TCP Transport APIs on QNX 6.5 Systems

Library Format	RTI TCP Transport Libraries[c]
Dynamic Release	libnddstransporttcp.so

[a]The libraries are in <NDDSHOME>/lib/<architecture>.

[b]These libraries are in <*openssl install dir*>/<architecture>/lib.

[c]These libraries are in <NDDSHOME>/lib/<architecture>.

Table 9.10 Additional Libraries for using RTI TCP Transport APIs on QNX 6.5 Systems

Library Format	RTI TCP Transport Libraries[a]
Dynamic Debug	libnddstransporttcpd.so
Static Release	libnddstransporttcpz.a
Static Debug	libnddstransporttcpzd.a

Table 9.11 Additional Libraries for using RTI TCP Transport APIs on QNX Systems with TLS Enabled

Library Format	RTI TLS Libraries[b]
Dynamic Release	libnddstls.so
Dynamic Debug	libnddstlsd.so
Static Release	libnddstlsz.a
Static Debug	libnddstlszd.a
OpenSSL Libraries	libssl.so libcrypto.so

9.12 Restarting Applications on QNX Systems

Due to a limitation in the POSIX API, if a process is unexpectedly interrupted in the middle of a critical section of code that is protected by a shared mutex semaphore, the OS is unable to automatically release the semaphore, making it impossible to reuse it by another application.

The Connext DDS shared-memory transport uses a shared mutex to protect access to the shared memory area across multiple processes.

It is possible under some extreme circumstances that if one application crashes or terminates ungracefully while executing code inside a critical section, the other applications sharing the same resource will not be able to continue their execution. If this situation occurs, you must manually delete the shared-memory mutex before re-launching any application in the same DDS domain.

---

[a]These libraries are in <NDDSHOME>/lib/<architecture>.

[b]These libraries are in <NDDSHOME>/lib/<architecture>.

Chapter 10 Solaris Platforms

Table 10.1 Supported Solaris Platforms lists the architectures supported on Solaris operating systems.

Table 10.1 Supported Solaris Platforms

Operating System	CPU	Compiler or Software Development Kit	RTI Architecture
Solaris 2.9	x86	gcc 3.3.2	i86Sol2.9gcc3.3.2
	UltraSPARC	CC 5.4 (Forte Dev 7, Sun One Studio 7)	sparcSol2.9cc5.4
Solaris 10	UltraSPARC	gcc3.4.2	sparcSol2.10gcc3.4.2
		Java Platform, Standard Edition JDK 1.7 or 1.8	
	UltraSPARC (with native 64-bit support)	gcc3.4.2	sparc64Sol2.10gcc3.4.2
		Java Platform, Standard Edition JDK 1.7 or 1.8	

Table 10.2 Building Instructions for Solaris Architectures lists the compiler flags and the libraries you will need to link into your application.

See also:

- VxWorks Platforms (Chapter 11 on page 109)
- Libraries Required for Using Monitoring (Section 10.9 on page 107)
- Libraries Required for using RTI Secure WAN Transport APIs (Section 10.10 on page 107)

Make sure you are consistent in your use of static, dynamic, debug and release versions of the libraries. Do not link both static and dynamic libraries. Similarly, do not mix release and debug libraries.

Table 10.3 Running Instructions for Solaris Architectures provides details on the environment variables that must be set at run time for a Solaris architecture.

When running on a Java 64-bit architecture, use the **-d64** flag in the command-line.

Table 10.4 Library-Creation Details for Solaris Architectures provides details on how the libraries were built by RTI. This table is provided strictly for informational purposes; you do not need to use these parameters to compile your application. You may find this information useful if you are involved in any in-depth debugging.

Table 10.2 Building Instructions for Solaris Architectures

API	Library Format	RTI Libraries or Jar Files[abc]	Required System Libraries	Required Compiler Flags
C	Static Release	libnddscz.a libnddscorez.a librticonnextmsgcz.a	sparc64Sol2.10gcc3.4.2: -ldl -lnsl -lsocket -lgen -lposix4 -lpthread -lm -lc All other architectures: -ldl -lnsl -lgenIO -lsocket -lgen -lposix4 -lpthread -lm -lc	sparc64Sol2.10gcc3.4.2: -DRTI_UNIX -m64 All other architectures: -DRTI_UNIX -m32
	Static Debug	libnddsczd.a libnddscorezd.a librticonnextmsgczd.a		
	Dynamic Release	libnddsc.so libnddscore.so librticonnextmsgc.so		
	Dynamic Debug	libnddscd.so libnddscored.so librticonnextmsgcd.so		

[a]Choose libnddscpp*.* for the Traditional C++ API or libnddscpp2*.* for the Modern C++ API.

[b]The RTI C/C++/Java libraries are in $(NDDSHOME)/lib/<architecture>. The jar files are in <NDDSHOME>/lib/java.

[c]The ***rticonnextmsg*** library only applies if you have the RTI Connext DDS Professional, Evaluation, or Basic package type. It is not provided with the RTI Connext DDS Core package type.

Table 10.2 Building Instructions for Solaris Architectures

API	Library Format	RTI Libraries or Jar Files[abc]	Required System Libraries	Required Compiler Flags
C++ (Traditional and Modern APIs)	Static Release	libnddscppz.a or libnddscpp2z.a libnddscz.a libnddscorez.a librticonnextmsgcppz.a	sparc64Sol2.10gcc3.4.2: -ldl -lnsl -lsocket -lgen -lposix4 -lpthread -lm -lc All other architectures: -ldl -lnsl -lgenIO -lsocket -lgen -lposix4 -lpthread -lm -lc	sparc64Sol2.10gcc3.4.2: -DRTI_UNIX -m64 All other architectures: -DRTI_UNIX -m32
	Static Debug	libnddscppzd.a or libnddscpp2zd.a libnddsczd.a libnddscorezd.a librticonnextmsgcppzd.a		
	Dynamic Release	libnddscpp.so or libnddscpp2.so libnddsc.so libnddscore.so librticonnextmsgcpp.so		
	Dynamic Debug	libnddscppd.so or libnddscpp2d.so libnddscd.so libnddscored.so librticonnextmsgcppd.so		
Java	Release	nddsjava.jar rticonnextmsg.jar	N/A	None required
	Debug	nddsjavad.jar rticonnextmsgd.jar		

[a]Choose libnddscpp*.* for the Traditional C++ API or libnddscpp2*.* for the Modern C++ API.

[b]The RTI C/C++/Java libraries are in $(NDDSHOME)/lib/<architecture>. The jar files are in <NDDSHOME>/lib/java.

[c]The ***rticonnextmsg*** library only applies if you have the RTI Connext DDS Professional, Evaluation, or Basic package type. It is not provided with the RTI Connext DDS Core package type.

Table 10.3 Running Instructions for Solaris Architectures

RTI Architecture	Library Format (Release & Debug)	Environment Variables
All supported Solaris architectures for Java	N/A	LD_LIBRARY_PATH= ${NDDSHOME}/lib/<*architecture*>: ${LD_LIBRARY_PATH} [a] **Note:** For all 64-bit Java architectures, use **-d64** in the command line.
All supported Solaris native architectures	Static	None required
	Dynamic	LD_LIBRARY_PATH= ${NDDSHOME}/lib/<*architecture*>: ${LD_LIBRARY_PATH} [b]

Table 10.4 Library-Creation Details for Solaris Architectures

RTI Architecture	Library Format	Compiler Flags Used by RTI
i86Sol2.9gcc3.3.2[c]	Static and Dynamic Release	-D_POSIX_C_SOURCE=199506L -D__EXTENSIONS__ -DSolaris2 -DSVR5 -DSUN4_SOLARIS2 -O -Wall -Wno-unknown-pragmas -fPIC -DPtrIntType=long -DCSREAL_IS_FLOAT -DCPU=i386 -DTARGET=\"i86Sol2.9gcc3.3.2\" -DNDEBUG -c -Wp,-MD -Wp
	Static and Dynamic Debug	-D_POSIX_C_SOURCE=199506L -D__EXTENSIONS__ -DSolaris2 -DSVR5 -DSUN4_SOLARIS2 -g -O -Wall -Wno-unknown-pragmas -fPIC -DPtrIntType=long -DCSREAL_IS_FLOAT -DCPU=i386 -DTARGET=\"i86Sol2.9gcc3.3.2\" -c -Wp,-MD -Wp

[a] $(NDDSHOME) is where Connext DDS is installed ${LD_LIBRARY_PATH} represents the value of the LD_LIBRARY_PATH variable prior to changing it to support Connext DDS. When using nddsjava.jar, the Java virtual machine (JVM) will attempt to load release versions of the native libraries. When using nddsjavad.jar, the JVM will attempt to load debug versions of the native libraries.

[b] $(NDDSHOME) is where Connext DDS is installed. ${LD_LIBRARY_PATH} represents the value of the LD_LIBRARY_PATH variable prior to changing it to support Connext DDS. When using nddsjava.jar, the Java virtual machine (JVM) will attempt to load release versions of the native libraries. When using nddsjavad.jar, the JVM will attempt to load debug versions of the native libraries.

[c]The C++ libnddscpp dynamic libraries were linked using g++; the C dynamic libraries, i.e. libnddscore and libnddsc, were linked using gcc.

Table 10.4 Library-Creation Details for Solaris Architectures

RTI Architecture	Library Format	Compiler Flags Used by RTI
sparcSol2.9cc5.4	Static and Dynamic Release	-D_POSIX_C_SOURCE=199506L -D__EXTENSIONS__ -DSolaris2 -DSVR5 -DSUN4_SOLARIS2 -KPIC -O +w -DPtrIntType=long -DCPU=SPARC -DTARGET=\"sparcSol2.9cc5.4\" -DNDEBUG -c
	Static and Dynamic Debug	-D_POSIX_C_SOURCE=199506L -D__EXTENSIONS__ -DSolaris2 -DSVR5 -DSUN4_SOLARIS2 -KPIC -g +w -DPtrIntType=long -DCPU=SPARC -DTARGET=\"sparcSol2.9cc5.4\" -c
sparcSol2.10gcc3.4.2[a]	Static and Dynamic Release	-D_POSIX_C_SOURCE=199506L -D__EXTENSIONS__ -DSolaris2 -DSVR5 -DSUN4_SOLARIS2 -O -Wall -Woverloaded-virtual -Wno-unknown-pragmas -DPtrIntType=long -DCSREAL_IS_FLOAT -DCPU=SPARC -DTARGET=\"sparcSol2.10gcc3.4.2\" -DNDEBUG -c -Wp, -MD
	Static and Dynamic Debug	-D_POSIX_C_SOURCE=199506L -D__EXTENSIONS__ -DSolaris2 -DSVR5 -DSUN4_SOLARIS2 -g -O -Wall -Woverloaded-virtual -Wno-unknown-pragmas -DPtrIntType=long -DCSREAL_IS_FLOAT -DCPU=SPARC -DTARGET=\"sparcSol2.10gcc3.4.2\" -c -Wp,-MD
sparc64Sol2.10gcc3.4.2[b]	Static and Dynamic Release	-m64 -fPIC -D_POSIX_C_SOURCE=199506L -D__EXTENSIONS__ -DSolaris2 -DSVR5 -DSUN4_SOLARIS2 -O -Wall -Wno-unknown-pragmas -DPtrIntType=long -DCSREAL_IS_FLOAT -DCPU=SPARC -DTARGET=\"sparc64Sol2.10gcc3.4.2\" -DNDEBUG -c -Wp, -MD
	Static and Dynamic Debug	-m64 -fPIC -D_POSIX_C_SOURCE=199506L -D__EXTENSIONS__ -DSolaris2 -DSVR5 -DSUN4_SOLARIS2 -g -O -Wall -Wno-unknown-pragmas -DPtrIntType=long -DCSREAL_IS_FLOAT -DCPU=SPARC -DTARGET=\"sparc64Sol2.10gcc3.4.2\" -c -Wp, -MD
All supported Solaris architectures for Java	Dynamic Release	-target 1.4 -source 1.4
	Dynamic Debug	-target 1.4 -source 1.4 -g

10.1 Request-Reply Communication Pattern

The Connext DDS Professional, Research, Basic, and Evaluation packages include support for the Request-Reply Communication Pattern, for all platforms in Table 10.1 Supported Solaris Platforms and all programming languages, except as noted below.

[a]The C++ libnddscpp dynamic libraries were linked using g++; the C dynamic libraries, i.e. libnddscore and libnddsc, were linked using gcc.

[b]The C++ libnddscpp dynamic libraries were linked using g++; the C dynamic libraries, i.e. libnddscore and libnddsc, were linked using gcc.

When using C++, the following platform does not support the Request-Reply Communication Pattern:

- sparcSol2.9cc5.4

10.2 Support for Modern C++ API

Connext DDS provides two different C++ APIs, which we refer to as the "Traditional C++" and "Modern C++" APIs. The Modern C++ API is only available for Solaris 2.10 platforms.

For more information on Modern C++, see "Traditional vs. Modern C++" in the User's Manual.

10.3 Multicast Support

Multicast is supported on Solaris platforms and is configured out of the box. That is, the default value for the initial peers list (**NDDS_DISCOVERY_PEERS**) includes a multicast address. See the online documentation for more information.

10.4 Supported Transports

Shared memory: Supported and enabled by default.

UDPv4: Supported and enabled by default.

UDPv6: Supported for all Solaris 2.9 and 2.10 platforms. The transport is not enabled by default, and the peers list must be modified to support IPv6. Traffic Class support is only provided for Solaris 2.10 platforms.

TCP/IPv4: Not supported.

10.4.1 Shared Memory Support

To see a list of shared memory resources in use, use the '**ipcs**' command. To clean up shared memory and shared semaphore resources, use the '**ipcrm**' command.

The shared memory keys used by Connext DDS are in the range of 0x400000. For example:

```
ipcs -m | grep 0x4
```

The shared semaphore keys used by Connext DDS are in the range of 0x800000; the shared mutex keys are in the range of 0xb00000. For example:

```
ipcs -s | grep 0x8

ipcs -s | grep 0xb
```

Please refer to the shared-memory transport online documentation for details on the shared memory and semaphore keys used by Connext DDS.

10.4.2 Increasing Available Shared Resources

Connext DDS uses System V semaphores to manage shared memory communication. If you plan to run multiple Connext DDS applications on the same node, at the same time, you may need to increase the number of available semaphores.

Each Connext DDS application that has shared memory enabled allocates 4 individual semaphores. The Solaris system defaults allow only 10 per host, which may not be enough (one is often used by the system, so you'll run out at the 3rd application).

To increase the number of semaphores available to Connext DDS, change the values of the following two parameters in **/etc/system**. (Starting in Solaris 10, there is an alternate mechanism to control these values, but changing **/etc/system** will also work.) The following values are just an example:

```
set semsys:seminfo_semmni = 100

set semsys:seminfo_semmns = 100
```

If these parameters already exist in **/etc/system**, change their values; otherwise, add the above lines to your **/etc/system** file.

> **WARNING:** Changing **/etc/system** should be done VERY carefully—incorrect editing of the file can render your system unbootable!

"System V" semaphores are allocated by creating groups of individual semaphores. The first parameter above controls the maximum number of semaphore groups and the second controls the maximum total number of semaphores (within any and all groups). Each Connext DDS application that has shared memory enabled allocates 4 groups of 1 semaphore each (per DDS domain). So setting the two values to the same number will work fine as far as Connext DDS is concerned. However, if other applications in the system want to allocate bigger groups, you could set "semsys:seminfo_semmns" larger than "semsys:seminfo_semmni." (Setting semmni bigger than semmns does not make any sense, since groups can't have less than 1 semaphore.)

In the absence of other applications using them, having 100 System V semaphores will allow you to use 25 domain ID/participant index combinations for Connext DDS applications. You probably will not need to increase the shared memory parameters, since the default allows 100 shared memory areas, enough for 50 applications.

10.5 Monotonic Clock Support

The monotonic clock (described in "Clock Selection" in the User's Manual) is supported on all Solaris platforms.

10.6 Thread Configuration

Table 10.5 Thread Settings for Solaris Platforms lists the thread settings for Solaris platforms.

Table 10.6 Thread-Priority Definitions for Solaris Platforms lists the thread-priority definitions.

10.6.1 Support for Controlling CPU Core Affinity for RTI Threads

Support for controlling CPU core affinity (described in "Controlling CPU Core Affinity" in the User's Manual) is not available for Solaris platforms.

Table 10.5 Thread Settings for Solaris Platforms

Applicable Thread	DDS_ ThreadSettings_ t	Platform-Specific Setting
Asynchronous Publisher, Asynchronous flushing thread	mask	OS default thread type
	priority	OS default thread priority
	stack_size	OS default thread stack size
	cpu_list	CPU core affinity not supported
	cpu_rotation	CPU core affinity not supported
Database thread	mask	DDS_THREAD_SETTINGS_STDIO
	priority	OS default thread priority
	stack_size	OS default thread stack size
	cpu_list	CPU core affinity not supported
	cpu_rotation	CPU core affinity not supported
Event thread	mask	DDS_THREAD_SETTINGS_STDIO | DDS_THREAD_ SETTINGS_FLOATING_POINT
	priority	OS default thread priority
	stack_size	OS default thread stack size
	cpu_list	CPU core affinity not supported
	cpu_rotation	CPU core affinity not supported

Table 10.5 Thread Settings for Solaris Platforms

Applicable Thread	DDS_ ThreadSettings_ t	Platform-Specific Setting
ReceiverPool threads	mask	DDS_THREAD_SETTINGS_STDIO | DDS_THREAD_ SETTINGS_FLOATING_POINT
	priority	OS default thread priority
	stack_size	OS default thread stack size
	cpu_list	CPU core affinity not supported
	cpu_rotation	CPU core affinity not supported

Table 10.6 Thread-Priority Definitions for Solaris Platforms

Thread-Priority Definition	Operating-System Priority
THREAD_PRIORITY_DEFAULT	If any of these constants are used to set the priority of the thread in the QoS, the OS's default thread priority will be used.
THREAD_PRIORITY_HIGH	
THREAD_PRIORITY_ABOVE_NORMAL	
THREAD_PRIORITY_NORMAL	
THREAD_PRIORITY_BELOW_NORMAL	
THREAD_PRIORITY_LOW	

10.7 Durable Writer History and Durable Reader State Features

The Durable Writer History and Durable Reader State features are not supported on Solaris platforms.

10.8 Distributed Logger Support

RTI Distributed Logger is not supported on Solaris platforms.

10.9 Libraries Required for Using Monitoring

Make sure you are consistent in your use of static, dynamic, debug and release versions of the libraries. For example, if your Connext DDS application is linked with the static release version of the Connext DDS libraries, you will need to also use the static release version of the monitoring library. Do not link both static and dynamic libraries. Similarly, do not mix release and debug libraries.

If you are statically linking your application with DDS libraries and you want to add monitoring to your application, you will also need to statically link the monitoring library. The library cannot be loaded dynamically strictly through the QoS profile because it also depends on DDS to publish its data. Therefore, it depends on DDS; the DDS functionality would cause duplicate symbols to be found resulting, in the termination of the process.

Note: If you plan to use *static* libraries, the RTI library from Table 10.7 Additional Libraries for Using Monitoring must appear *first* in the list of libraries to be linked.

Table 10.7 Additional Libraries for Using Monitoring

Static Release	Static Debug	Dynamic Release	Dynamic Debug
librtimonitoringz.a	librtimonitoringzd.a	librtimonitoring.so	librtimonitoringd.so

10.10 Libraries Required for using RTI Secure WAN Transport APIs

This section is only relevant if you have installed RTI Secure WAN Transport. This feature is not part of the standard Connext DDS package. If you choose to use it, it must be downloaded and installed separately. It is only available on specific architectures. See the RTI Secure WAN Transport Release Notes and RTI Secure WAN Transport Installation Guide for details.

To use the Secure WAN Transport APIs, link against the additional libraries in Table 10.8 Additional Libraries for using RTI Secure WAN Transport APIs. (Select the files appropriate for your chosen library format.)

Table 10.8 Additional Libraries for using RTI Secure WAN Transport APIs

Library Format	RTI Secure WAN Transport Libraries[a]	OpenSSL Libraries[b]
Dynamic Release	libnddstransportwan.so libnddstransporttls.so	libssl.a libcrypto.a
Dynamic Debug	libnddstransportwand.so libnddstransporttlsd.so	
Static Release	libnddstransporttlsz.a libnddstransporttlszd.a	
Static Debug	libnddstransportwanz.a libnddstransportwanzd.a	

[a]These libraries are in <NDDSHOME>/lib/<architecture>.

[b]These libraries are in <openssl install dir>/<architecture>/lib, where <openssl install dir> is where OpenSSL is i.

Chapter 11 VxWorks Platforms

Table 11.1 Supported VxWorks Target Platforms lists the architectures supported on VxWorks operating systems. You can build a VxWorks application by cross-compiling from your development host.

Table 11.1 Supported VxWorks Target Platforms

Operating System	CPU	Compiler	RTI Architecture[a]
VxWorks 5.5	PPC 603	gcc 2.96	ppc603Vx5.5gcc
	PPC 604	gcc 2.96	ppc604Vx5.5gcc
	PPC 750	gcc 2.96	ppc603Vx5.5gcc
	PPC 7400	gcc 2.96	ppc603Vx5.5gcc
VxWorks 6.3, 6.4	Any PowerPC CPU with floating-point hardware that is backwards-compatible with 32-bit PowerPC 604[b]	gcc 3.4.4	For kernel modules: ppc604Vx6.3gcc3.4.4 For Real Time Processes: ppc604Vx6.3gcc3.4.4_rtp
VxWorks 6.5			For kernel modules: ppc604Vx6.5gcc3.4.4 For Real Time Processes: ppc604Vx6.5gcc3.4.4_rtp

[a] For use with Windows and/or Solaris Hosts as supported by Wind River Systems.

[b] Some PowerPC cores such as e500v1 and e500v2 are not fully backwards-compatible with PPC 604.

Table 11.1 Supported VxWorks Target Platforms

Operating System	CPU	Compiler	RTI Architecture[a]
VxWorks 6.6	Pentium	gcc 4.1.2	For Kernel Modules: pentiumVx6.6gcc4.1.2 For Real Time Processes: pentiumVx6.6gcc4.1.2_rtp
	Any PowerPC CPU with floating-point hardware that is backwards-compatible with 32-bit PowerPC 604[b]	gcc 4.1.2	For Kernel Modules: ppc604Vx6.6gcc4.1.2 For Real Time Processes: ppc604Vx6.6gcc4.1.2_rtp
	PPC 405[c]	gcc 4.1.2	For Kernel Modules: ppc405Vx6.6gcc4.1.2 For Real Time Processes: ppc405Vx6.6gcc4.1.2_rtp
VxWorks 6.7	Pentium	gcc 4.1.2	For Kernel Modules: pentiumVx6.7gcc4.1.2 For Real Time Processes: pentiumVx6.7gcc4.1.2_rtp
	Any PowerPC CPU with floating-point hardware that is backwards-compatible with 32-bit PowerPC 604[d]	gcc 4.1.2	For Kernel Modules: ppc604Vx6.7gcc4.1.2 For Real Time Processes on non-SMP systems: ppc604Vx6.7gcc4.1.2_rtp For Real Time Processes on SMP systems: ppc604Vx6.7gcc4.1.2_smp
	PPC 405[e]	gcc 4.1.2	For Kernel Modules: ppc405Vx6.7gcc4.1.2 For Real Time Processes: ppc405Vx6.7gcc4.1.2_rtp

[a]For use with Windows and/or Solaris Hosts as supported by Wind River Systems.

[b]Some PowerPC cores such as e500v1 and e500v2 are not fully backwards-compatible with PPC 604.

[c]For ppc405, the architecture string is the same for VxWorks 6.6 and 6.7.

[d]Some PowerPC cores such as e500v1 and e500v2 are not fully backwards-compatible with PPC 604.

[e]For ppc405, the architecture string is the same for VxWorks 6.6 and 6.7.

Table 11.1 Supported VxWorks Target Platforms

Operating System	CPU	Compiler	RTI Architecture[a]
VxWorks 6.8	Pentium	gcc 4.1.2	For Kernel Modules: pentiumVx6.8gcc4.1.2 For Real Time Processes: pentiumVx6.8gcc4.1.2_rtp
	Any PowerPC CPU with floating-point hardware that is backwards-compatible with 32-bit PowerPC 604	gcc 4.1.2	For Kernel Modules: ppc604Vx6.8gcc4.1.2 For Real Time Processes: ppc604Vx6.8gcc4.1.2_rtp
VxWorks 6.9	Pentium32-bit	gcc 4.3.3	For Kernel Modules: pentiumVx6.9gcc4.3.3 For Real Time Processes: pentiumVx6.9gcc4.3.3_rtp
	Any PowerPC CPU with floating-point hardware that is backwards-compatible with 32-bit PowerPC 604	gcc 4.3.3	For Kernel Modules: ppc604Vx6.9gcc4.3.3 For Real Time Processes: ppc604Vx6.9gcc4.3.3_rtp
VxWorks 6.9.3.2	x64	gcc 4.3.3	For Kernel Modules: pentium64Vx6.9gcc4.3.3 For Real Time Processes: pentium64Vx6.9gcc4.3.3_rtp
VxWorks 6.9.4	PPC (e500v2)	gcc 4.3.3	For Kernel Modules: ppce500v2Vx6.9.4gcc4.3.3 For Real-Time Processes: ppce500v2Vx6.9.4gcc4.3.3_rtp
VxWorks 7.0	Pentium (32 bit)	gcc 4.3.3	For Kernel Modules: pentiumVx7.0gcc4.3.3 For Real Time Processes: pentiumVx7.0gcc4.3.3_rtp
VxWorks 653 2.3	sbc8641d	gcc 3.3.2	sbc8641Vx653-2.3gcc3.3.2
	SIMPC	gcc 3.3.2	simpcVx653-2.3gcc3.3.2

[a]For use with Windows and/or Solaris Hosts as supported by Wind River Systems.

The following tables list the libraries you will need to link into your application and the required compiler flags:

- Table 11.2 Building Instructions for VxWorks 5.x - 7.x Architectures
- Table 11.3 Building Instructions for VxWorks 653 Architectures

See also:

- Libraries Required for Using Distributed Logger (Section 11.12 on page 131)
- Libraries Required for Using Monitoring (Section 11.13 on page 132)

Make sure you are consistent in your use of static, dynamic, debug and release versions of the libraries. Do not link both static and dynamic libraries. Similarly, do not mix release and debug libraries.

Compiling a Connext DDS application for VxWorks depends on the development platform. For more information, such as specific compiler flags, see the *VxWorks Programmer's Guide*. Table 11.4 Library-Creation Details for All VxWorks Architectures provides details on how the VxWorks libraries were built. We recommend that you use similar settings.

Cross-compiling for any VxWorks platform is similar to building for a UNIX target. To build a VxWorks application, create a makefile that reflects the compiler and linker for your target with appropriate flags defined. There will be several target-specific compile flags you must set to build correctly. For more information, see the *VxWorks Programmer's Guide*.

Table 11.2 Building Instructions for VxWorks 5.x - 7.x Architectures

API	Library Format	Required RTI Libraries[abc]	Required Kernel Components	Required Compiler Flags
C++ (Traditional and Modern APIs)	Static Release	libnddscppz.a or libnddscpp2z.a libnddscz.a libnddscorez.a librticonnextmsgcppz.a	INCLUDE_TIMESTAMP For VxWorks 6.4 and below, also use: INCLUDE_ZBUF_SOCK INCLUDE_IGMP For VxWorks 6.3 and higher, also use: INCLUDE_POSIX_CLOCKS For RTI architectures with SMP support for VxWorks 6.7 and higher[d], also use: INCLUDE_TLS	-DRTI_VXWORKS
	Static Debug	libnddscppzd.a or libnddscpp2zd.a libnddsczd.a libnddscorezd.a librticonnextmsgcppzd.a		
	Dynamic Release	libnddscpp2.so (for RTP mode) libnddscpp2.lo (for kernel mode) (not supported for VxWorks 5.x) librticonnextmsgcpp.so (for RTP mode) librticonnextmsgcpp.lo (for kernel mode) (not supported for VxWorks 5.x) libnddsc.so libnddscore.so libnddscpp.so		

[a]Choose libnddscpp*.* for the Traditional C++ API or libnddscpp2*.* for the Modern C++ API.

[b]The Connext DDS C/C++ libraries are in <NDDSHOME>/lib/<*architecture*>.

[c]The ***rticonnextmsg*** library only applies if you have the RTI Connext DDS Professional, Evaluation, or Basic package type. It is not provided with the RTI Connext DDS Core package type.

[d]In this version, only ppc604Vx6.7gcc4.1.2_smp.

Table 11.2 Building Instructions for VxWorks 5.x - 7.x Architectures

API	Library Format	Required RTI Libraries[abc]	Required Kernel Components	Required Compiler Flags
	Dynamic Debug	libnddscpp2d.so (for RTP mode) libnddscpp2d.lo (for kernel mode) (not supported for VxWorks 5.x) librticonnextmsgcppd.so (for RTP mode) librticonnextmsgcppd.lo (for kernel mode) (not supported for VxWorks 5.x) libnddscd.so libnddscored.so libnddscppd.so		
C	Static Release	libnddscz.a libnddscorez.a librticonnextmsgcz.a	INCLUDE_TIMESTAMP For VxWorks 6.4 and below, also use: INCLUDE_ZBUF_SOCK INCLUDE_IGMP For VxWorks 6.3 and higher, also use: INCLUDE_POSIX_CLOCKS For RTI architectures with SMP support for VxWorks 6.7 and higher[d], also use: INCLUDE_TLS	-DRTI_VXWORKS
	Static Debug	libnddsczd.a libnddscorezd.a librticonnextmsgczd.a		
	Dynamic Release	libnddsc.so libnddscore.so librticonnextmsgc.so		
	Dynamic Debug	libnddscd.so libnddscored.so librticonnextmsgcd.so		

[a]Choose libnddscpp*.* for the Traditional C++ API or libnddscpp2*.* for the Modern C++ API.

[b]The Connext DDS C/C++ libraries are in <NDDSHOME>/lib/<*architecture*>.

[c]The ***rticonnextmsg*** library only applies if you have the RTI Connext DDS Professional, Evaluation, or Basic package type. It is not provided with the RTI Connext DDS Core package type.

[d]In this version, only ppc604Vx6.7gcc4.1.2_smp.

Table 11.3 Building Instructions for VxWorks 653 Architectures

API	Library Format	Required RTI Libraries[a]	Required Kernel Components	Required Compiler Flags
C++ (Traditional API)	Static Release	libnddscppz.a libnddscz.a libnddscorez.a librticonnextmsgcppz.a	See either: Table 11.11 Required Kernel Components for sbc8641Vx653-2.3gcc3.3.2 or Table 11.12 Required Kernel Components for simpcVx653-2.3gcc3.3.2	-DRTI_VXWORKS -DRTI_VX653
	Static Debug	libnddscppzd.a libnddsczd.a libnddscorezd.a librticonnextmsgcppzd.a		
	Dynamic Release	libnddscpp.so libnddsc.so libnddscore.so librticonnextmsgcpp.so		
	Dynamic Debug	libnddscppd.so libnddscd.so libnddscored.so librticonnextmsgcppd.so		
C	Static Release	libnddscz.a libnddscorez.a librticonnextmsgcz.a	See either: Table 11.11 Required Kernel Components for sbc8641Vx653-2.3gcc3.3.2 or Table 11.12 Required Kernel Components for simpcVx653-2.3gcc3.3.2	-DRTI_VXWORKS -DRTI_VX653
	Static Debug	libnddsczd.a libnddscorezd.a librticonnextmsgczd.a		
	Dynamic Release	libnddsc.so libnddscore.so librticonnextmsgc.so		
	Dynamic Debug	libnddscd.so libnddscored.so librticonnextmsgcd.so		

[a]The Connext DDS C/C++ libraries are in <NDDSHOME>/lib/<architecture>.

Table 11.4 Library-Creation Details for All VxWorks Architectures

RTI Architecture	Library Format	Compiler Flags Used by RTI
pentium64Vx6.9gcc4.3.3	Static or Dynamic Release	ccpentium -march=x86-64 -m64 -mcmodel=small -mno-red-zone -fno-builtin -ansi -TOOL_ FAMILY=gnu -DTOOL=gnu -D_WRS_KERNEL -D__PROTOTYPE_5_0 -O -DRTI_64BIT -DRTI_X64CPU -DVXWORKS_MAJOR_VERSION=6 -DVXWORKS_MINOR_ VERSION=9 -Wall -Wno-unknown-pragmas -DRTS_VXWORKS -DPtrIntType=long -DCSREAL_IS_FLOAT -DNDEBUG -Wp,-MD
	Static or Dynamic Debug	ccpentium -march=x86-64 -m64 -mcmodel=small -mno-red-zone -fno-builtin -ansi -TOOL_ FAMILY=gnu -DTOOL=gnu -D_WRS_KERNEL -D__PROTOTYPE_5_0 -g -DRTI_64BIT -DRTI_X64CPU -DVXWORKS_MAJOR_VERSION=6 -DVXWORKS_MINOR_ VERSION=9 -Wall -Wno-unknown-pragmas -DRTS_VXWORKS -DPtrIntType=long -DCSREAL_IS_FLOAT -Wp,-MD
pentium64Vx6.9gcc4.3.3_ rtp	Static or Dynamic Release	ccpentium -march=x86-64 -m64 -mcmodel=small -mno-red-zone -fno-builtin -ansi -mrtp -TOOL_FAMILY=gnu -DTOOL=gnu -D_WRS_KERNEL -D__PROTOTYPE_5_0 -O -DRTI_64BIT -DRTI_X64CPU -DVXWORKS_MAJOR_VERSION=6 -DVXWORKS_ MINOR_VERSION=9 -Wall -Wno-unknown-pragmas -DRTS_VXWORKS -DPtrIntType=long -DCSREAL_IS_FLOAT -DNDEBUG -Wp,-MD
	Static or Dynamic Debug	ccpentium -march=x86-64 -m64 -mcmodel=small -mno-red-zone -fno-builtin -ansi -mrtp -TOOL_FAMILY=gnu -DTOOL=gnu -D_WRS_KERNEL -D__PROTOTYPE_5_0 -g -DRTI_64BIT -DRTI_X64CPU -DVXWORKS_MAJOR_VERSION=6 -DVXWORKS_ MINOR_VERSION=9 -Wall -Wno-unknown-pragmas -DRTS_VXWORKS -DPtrIntType=long -DCSREAL_IS_FLOAT -Wp,-MD
pentiumVx6.6gcc4.1.2	Static or Dynamic Release	-march=pentium -fno-builtin -ansi -DTOOL=gnu -D_WRS_KERNEL -D_PROTOTYPE_5_0 -DVXWORKS_MAJOR_VERSION=6 -DVXWORKS_MINOR_VERSION=6 -O -Wall -Wno-unknown-pragmas -DRTS_VXWORKS -DTOOL_FAMILY=gnu -DPtrIntType=long -DCPU=PENTIUM -DNDEBUG -c -Wp,-MD
	Static or Dynamic Debug	-march=pentium -fno-builtin -ansi -DTOOL=gnu -D_WRS_KERNEL -D_PROTOTYPE_5_0 -g -DVXWORKS_MAJOR_VERSION=6 -DVXWORKS_MINOR_VERSION=6 -O -Wall -Wno-unknown-pragmas -DRTS_VXWORKS -DTOOL_FAMILY=gnu -DPtrIntType=long -DCPU=PENTIUM -c -Wp,-MD

Table 11.4 Library-Creation Details for All VxWorks Architectures

RTI Architecture	Library Format	Compiler Flags Used by RTI
pentiumVx6.6gcc4.1.2_rtp	Static Release	-march=i486 -ansi -DTOOL=gnu -mrtp -D_PROTOTYPE_5_0 -DVXWORKS_MAJOR_VERSION=6 -DVXWORKS_MINOR_VERSION=6 -O -Wall -Wno-unknown-pragmas -DRTS_VXWORKS -DTOOL_FAMILY=gnu -DPtrIntType=long -DCPU=PENTIUM -DNDEBUG -c -Wp,-MD
	Static Debug	-march=i486 -ansi -DTOOL=gnu -mrtp -D_PROTOTYPE_5_0 -g -DVXWORKS_MAJOR_VERSION=6 -DVXWORKS_MINOR_VERSION=6 -O -Wall -Wno-unknown-pragmas -DRTS_VXWORKS -DTOOL_FAMILY=gnu -DPtrIntType=long -DCPU=PENTIUM -c -Wp,-MD
	Dynamic Release	-march=i486 -ansi -DTOOL=gnu -mrtp -D_PROTOTYPE_5_0 -fPIC -DVXWORKS_MAJOR_VERSION=6 -DVXWORKS_MINOR_VERSION=6 -O -Wall -Wno-unknown-pragmas -DRTS_VXWORKS -DTOOL_FAMILY=gnu -DPtrIntType=long -DCPU=PENTIUM -DNDEBUG -c -Wp,-MD
	Dynamic Debug	-march=i486 -ansi -DTOOL=gnu -mrtp -D_PROTOTYPE_5_0 -fPIC -g -DVXWORKS_MAJOR_VERSION=6 -DVXWORKS_MINOR_VERSION=6 -O -Wall -Wno-unknown-pragmas -DRTS_VXWORKS -DTOOL_FAMILY=gnu -DPtrIntType=long -DCPU=PENTIUM -c -Wp,-MD
pentiumVx6.7gcc4.1.2	Static or Dynamic Release	-march=pentium -fno-builtin -ansi -DTOOL=gnu -D_WRS_KERNEL -D_PROTOTYPE_5_0 -DVXWORKS_MAJOR_VERSION=6 -DVXWORKS_MINOR_VERSION=7 -O -Wall -Wno-unknown-pragmas -DRTS_VXWORKS -DTOOL_FAMILY=gnu -DPtrIntType=long -DCPU=PENTIUM -DNDEBUG -c -Wp,-MD
	Static or Dynamic Debug	-march=pentium -fno-builtin -ansi -DTOOL=gnu -D_WRS_KERNEL -D_PROTOTYPE_5_0 -g -DVXWORKS_MAJOR_VERSION=6 -DVXWORKS_MINOR_VERSION=7 -O -Wall -Wno-unknown-pragmas -DRTS_VXWORKS -DTOOL_FAMILY=gnu -DPtrIntType=long -DCPU=PENTIUM -c -Wp,-MD

Table 11.4 Library-Creation Details for All VxWorks Architectures

RTI Architecture	Library Format	Compiler Flags Used by RTI
pentiumVx6.7gcc4.1.2_rtp	Static Release	-march=i486 -ansi -DTOOL=gnu -mrtp -D_PROTOTYPE_5_0 -DVXWORKS_MAJOR_VERSION=6 -DVXWORKS_MINOR_VERSION=7 -O -Wall -Wno-unknown-pragmas -DRTS_VXWORKS -DTOOL_FAMILY=gnu -DPtrIntType=long -DCPU=PENTIUM -DNDEBUG -c -Wp,-MD
	Static Debug	-march=i486 -ansi -DTOOL=gnu -mrtp -D_PROTOTYPE_5_0 -g -DVXWORKS_MAJOR_VERSION=6 -DVXWORKS_MINOR_VERSION=7 -O -Wall -Wno-unknown-pragmas -DRTS_VXWORKS -DTOOL_FAMILY=gnu -DPtrIntType=long -DCPU=PENTIUM -c -Wp,-MD
	Dynamic Release	-march=i486 -ansi -DTOOL=gnu -mrtp -D_PROTOTYPE_5_0 -fPIC -DVXWORKS_MAJOR_VERSION=6 -DVXWORKS_MINOR_VERSION=7 -O -Wall -Wno-unknown-pragmas -DRTS_VXWORKS -DTOOL_FAMILY=gnu -DPtrIntType=long -DCPU=PENTIUM -DNDEBUG -c -Wp,-MD
	Dynamic Debug	-march=i486 -ansi -DTOOL=gnu -mrtp -D_PROTOTYPE_5_0 -fPIC -g -DVXWORKS_MAJOR_VERSION=6 -DVXWORKS_MINOR_VERSION=7 -O -Wall -Wno-unknown-pragmas -DRTS_VXWORKS -DTOOL_FAMILY=gnu -DPtrIntType=long -DCPU=PENTIUM -c -Wp,-MD
pentiumVx6.8gcc4.1.2	Static or Dynamic Release	ccpentium -m32 -march=pentium -fno-builtin -ansi -DCPU=PENTIUM -DTOOL_FAMILY=gnu -DTOOL=gnu -D_WRS_KERNEL -D__PROTOTYPE_5_0 -O -DVXWORKS_MAJOR_VERSION=6 -DVXWORKS_MINOR_VERSION=8 -Wall -Wno-unknown-pragmas -DRTS_VXWORKS -DPtrIntType=long -DCSREAL_IS_FLOAT -DNDEBUG -Wp,-MD
	Static or Dynamic Debug	ccpentium -m32 -march=pentium -fno-builtin -ansi -DCPU=PENTIUM -DTOOL_FAMILY=gnu -DTOOL=gnu -D_WRS_KERNEL -D__PROTOTYPE_5_0 -g -DVXWORKS_MAJOR_VERSION=6 -DVXWORKS_MINOR_VERSION=8 -Wall -Wno-unknown-pragmas -DRTS_VXWORKS -DPtrIntType=long -DCSREAL_IS_FLOAT -Wp,-MD
pentiumVx6.8gcc4.1.2_rtp	Static or Dynamic Release	ccpentium -m32 -march=pentium -ansi -DCPU=PENTIUM -DTOOL_FAMILY=gnu -DTOOL=gnu -mrtp -D__PROTOTYPE_5_0 -O -Wall -Wno-unknown-pragmas -DRTS_VXWORKS -DVXWORKS_MAJOR_VERSION=6 -DVXWORKS_MINOR_VERSION=8 -DPtrIntType=long -DCSREAL_IS_FLOAT -DNDEBUG -Wp,-MD
	Static or Dynamic Debug	ccpentium -m32 -march=pentium -ansi -DCPU=PENTIUM -DTOOL_FAMILY=gnu -DTOOL=gnu -mrtp -D__PROTOTYPE_5_0 -g -Wall -Wno-unknown-pragmas -DRTS_VXWORKS -DVXWORKS_MAJOR_VERSION=6 -DVXWORKS_MINOR_VERSION=8 -DPtrIntType=long -DCSREAL_IS_FLOAT -Wp,-MD

Table 11.4 Library-Creation Details for All VxWorks Architectures

RTI Architecture	Library Format	Compiler Flags Used by RTI
pentiumVx6.9gcc4.3.3	Static or Dynamic Release	ccpentium -m32 -march=pentium -fno-builtin -ansi -DCPU=PENTIUM -DTOOL_FAMILY=gnu -DTOOL=gnu -D_WRS_KERNEL -D__PROTOTYPE_5_0 -O -DVXWORKS_MAJOR_VERSION=6 -DVXWORKS_MINOR_VERSION=9 -Wall -Wno-unknown-pragmas -DRTS_VXWORKS -DPtrIntType=long -DCSREAL_IS_FLOAT -DNDEBUG -Wp,-MD
	Static or Dynamic Debug	ccpentium -m32 -march=pentium -fno-builtin -ansi -DCPU=PENTIUM -DTOOL_FAMILY=gnu -DTOOL=gnu -D_WRS_KERNEL -D__PROTOTYPE_5_0 -g -DVXWORKS_MAJOR_VERSION=6 -DVXWORKS_MINOR_VERSION=9 -Wall -Wno-unknown-pragmas -DRTS_VXWORKS -DPtrIntType=long -DCSREAL_IS_FLOAT -DCPU=PENTIUM -Wp,-MD
pentiumVx6.9gcc4.3.3_rtp	Static or Dynamic Release	ccpentium -m32 -march=pentium -ansi -DCPU=PENTIUM -DTOOL_FAMILY=gnu -DTOOL=gnu -mrtp -D__PROTOTYPE_5_0 -O -Wall -Wno-unknown-pragmas -DRTS_VXWORKS -DVXWORKS_MAJOR_VERSION=6 -DVXWORKS_MINOR_VERSION=9 -DPtrIntType=long -DCSREAL_IS_FLOAT -DNDEBUG -Wp,-MD
	Static or Dynamic Debug	ccpentium -m32 -march=pentium -ansi -DCPU=PENTIUM -DTOOL_FAMILY=gnu -DTOOL=gnu -mrtp -D__PROTOTYPE_5_0 -g -Wall -Wno-unknown-pragmas -DRTS_VXWORKS -DVXWORKS_MAJOR_VERSION=6 -DVXWORKS_MINOR_VERSION=9 -DPtrIntType=long -DCSREAL_IS_FLOAT -Wp,-MD
pentiumVx7.0gcc4.3.3	Static or Dynamic Release	ccpentium -mtune=pentium -march=pentium -nostdlib -fno-builtin -fno-defer-pop -fno-implicit-fp -ansi -fno-zero-initialized-in-bss -Wall -MD -MP -DCPU=_VX_PENTIUM4 -DTOOL_FAMILY=gnu -DTOOL=gnu -D_WRS_KERNEL -g -DVXWORKS_MAJOR_VERSION=7 -DVXWORKS_MINOR_VERSION=0 -Wall -Wno-unknown-pragmas -DRTS_VXWORKS -DPtrIntType=long -DCSREAL_IS_FLOAT -DNDEBUG
	Static or Dynamic Debug	ccpentium -mtune=pentium -march=pentium -nostdlib -fno-builtin -fno-defer-pop -fno-implicit-fp -ansi -fno-zero-initialized-in-bss -Wall -MD -MP -DCPU=_VX_PENTIUM4 -DTOOL_FAMILY=gnu -DTOOL=gnu -D_WRS_KERNEL -g -DVXWORKS_MAJOR_VERSION=7 -DVXWORKS_MINOR_VERSION=0 -Wall -Wno-unknown-pragmas -DRTS_VXWORKS -DPtrIntType=long -DCSREAL_IS_FLOAT
pentiumVx7.0gcc4.3.3_rtp	Static or Dynamic Release	ccpentium -mtune=pentium4 -march=pentium4 -mrtp -fno-strict-aliasing -fasm -Wall -MD -MP -D_VX_CPU=_VX_PENTIUM -D_VX_TOOL_FAMILY=gnu -D_VX_TOOL=gnu -D_C99 -D_HAS_C9X -std=c99 -g -Wall -Wno-unknown-pragmas -DRTS_VXWORKS -DVXWORKS_MAJOR_VERSION=7 -DVXWORKS_MINOR_VERSION=0 -DPtrIntType=long -DCSREAL_IS_FLOAT -DNDEBUG
	Static or Dynamic Debug	ccpentium -mtune=pentium4 -march=pentium4 -mrtp -fno-strict-aliasing -fasm -Wall -MD -MP -D_VX_CPU=_VX_PENTIUM -D_VX_TOOL_FAMILY=gnu -D_VX_TOOL=gnu -D_C99 -D_HAS_C9X -std=c99 -g -Wall -Wno-unknown-pragmas -DRTS_VXWORKS -DVXWORKS_MAJOR_VERSION=7 -DVXWORKS_MINOR_VERSION=0 -DPtrIntType=long -DCSREAL_IS_FLOAT -DNDEBUG

Table 11.4 Library-Creation Details for All VxWorks Architectures

RTI Architecture	Library Format	Compiler Flags Used by RTI
ppc405Vx6.6gcc4.1.2	Static or Dynamic Release	-mcpu=405 -fno-builtin -mlongcall -DTOOL=gnu -mstrict-align -msoft-float -ansi -D_WRS_KERNEL -D_PROTOTYPE_5_0 -DVXWORKS_MAJOR_VERSION=6 -DVXWORKS_MINOR_VERSION=6 -O -Wall -Wno-unknown-pragmas -DRTS_VXWORKS -DTOOL=gnu -DTOOL_FAMILY=gnu -DPtrIntType=long -DCPU=PPC405 -DNDEBUG -c -Wp,-MD
	Static or Dynamic Debug	-mcpu=405 -fno-builtin -mlongcall -DTOOL=gnu -mstrict-align -msoft-float -ansi -D_WRS_KERNEL -D_PROTOTYPE_5_0 -g -DVXWORKS_MAJOR_VERSION=6 -DVXWORKS_MINOR_VERSION=6 -O -Wall -Wno-unknown-pragmas -DRTS_VXWORKS -DTOOL=gnu -DTOOL_FAMILY=gnu -DPtrIntType=long -DCPU=PPC405 -c -Wp,-MD
ppc405Vx6.6gcc4.1.2_rtp	Static Release	-msoft-float -mlongcall -mregnames -mstrict-align -ansi -DTOOL=gnu -mrtp -D_PROTOTYPE_5_0 -DVXWORKS_MAJOR_VERSION=6 -DVXWORKS_MINOR_VERSION=6 -O -Wall -Wno-unknown-pragmas -DRTS_VXWORKS -DTOOL=sfgnu -DTOOL_FAMILY=gnu -DPtrIntType=long -DCPU=PPC32 -DNDEBUG -c -Wp,-MD
	Static Debug	-msoft-float -mlongcall -mregnames -mstrict-align -ansi -DTOOL=gnu -mrtp -fPIC -shared -D_PROTOTYPE_5_0 -g -DVXWORKS_MAJOR_VERSION=6 -DVXWORKS_MINOR_VERSION=6 -O -Wall -Wno-unknown-pragmas -DRTS_VXWORKS -DTOOL=sfgnu -DTOOL_FAMILY=gnu -DPtrIntType=long -DCPU=PPC32 -c -Wp,-MD
ppc603Vx5.5gcc	Static or Dynamic Release	-mcpu=603 -G 0 -fno-builtin -mlongcall -D__PROTOTYPE_5_0 -DVXWORKS_MAJOR_VERSION=5 -DVXWORKS_MINOR_VERSION=5 -O -Wall -Wno-unknown-pragmas -DRTS_VXWORKS -DPtrIntType=long -DCSREAL_IS_FLOAT -DCPU=PPC603 -DNDEBUG -c -Wp,-MD
	Static or Dynamic Debug	-mcpu=603 -G 0 -fno-builtin -mlongcall -D__PROTOTYPE_5_0 -g-DVXWORKS_MAJOR_VERSION=5 -DVXWORKS_MINOR_VERSION=5 -O -Wall -Wno-unknown-pragmas -DRTS_VXWORKS -DPtrIntType=long -DCSREAL_IS_FLOAT -DCPU=PPC603 -c -Wp,-MD
ppc604Vx5.5gcc	Static or Dynamic Release	-mcpu=604 -G 0 -fno-builtin -mlongcall -D__PROTOTYPE_5_0 -DVXWORKS_MAJOR_VERSION=5 -DVXWORKS_MINOR_VERSION=5 -O -Wall -Wno-unknown-pragmas -DRTS_VXWORKS -DPtrIntType=long -DCPU=PPC604 -DNDEBUG -c -Wp,-MD
	Static or Dynamic Debug	-mcpu=604 -G 0 -fno-builtin -mlongcall -D__PROTOTYPE_5_0 -g -DVXWORKS_MAJOR_VERSION=5 -DVXWORKS_MINOR_VERSION=5 -O -Wall -Wno-unknown-pragmas -DRTS_VXWORKS -DPtrIntType=long -DCPU=PPC604 -c -Wp,-MD

Table 11.4 Library-Creation Details for All VxWorks Architectures

RTI Architecture	Library Format	Compiler Flags Used by RTI
ppc604Vx6.3gcc3.4.4	Static or Dynamic Release	-mcpu=604 -fno-builtin -mlongcall -DTOOL=gnu -mstrict-align -mno-implicit-fp -ansi -D_WRS_KERNEL -D_PROTOTYPE_5_0 -DVXWORKS_MAJOR_VERSION=6 -DVXWORKS_MINOR_VERSION=3 -O -Wall -Wno-unknown-pragmas -DRTS_VXWORKS -DTOOL_FAMILY=gnu -DPtrIntType=long -DCPU=PPC604 -DNDEBUG -c -Wp,-MD
	Static or Dynamic Debug	-mcpu=604 -fno-builtin -mlongcall -DTOOL=gnu -mstrict-align -mno-implicit-fp -ansi -D_WRS_KERNEL -D_PROTOTYPE_5_0 -g -DVXWORKS_MAJOR_VERSION=6 -DVXWORKS_MINOR_VERSION=3 -O -Wall -Wno-unknown-pragmas -DRTS_VXWORKS -DTOOL_FAMILY=gnu -DPtrIntType=long -DCPU=PPC604 -c -Wp,-MD
ppc604Vx6.3gcc3.4.4_rtp	Static Release	-mhard-float -mlongcall -mregnames -mstrict-align -ansi -DTOOL=gnu -mrtp -D_PROTOTYPE_5_0 -DVXWORKS_MAJOR_VERSION=6 -DVXWORKS_MINOR_VERSION=3 -O -Wall -Wno-unknown-pragmas -DRTS_VXWORKS -DTOOL_FAMILY=gnu -DPtrIntType=long -DCPU=PPC32 -DNDEBUG -c -Wp,-MD
	Static Debug	-mhard-float -mlongcall -mregnames -mstrict-align -ansi -DTOOL=gnu -mrtp -fPIC -shared -D_PROTOTYPE_5_0 -g -DVXWORKS_MAJOR_VERSION=6 -DVXWORKS_MINOR_VERSION=3 -O -Wall -Wno-unknown-pragmas -DRTS_VXWORKS -DTOOL_FAMILY=gnu -DPtrIntType=long -DCPU=PPC32 -c -Wp,-MD
ppc604Vx6.5gcc3.4.4	Static or Dynamic Release	-mcpu=604 -mstrict-align -fno-builtin -ansi -mlongcall -mno-implicit-fp -D_WRS_KERNEL -D_PROTOTYPE_5_0 -DVXWORKS_MAJOR_VERSION=6 -DVXWORKS_MINOR_VERSION=5 -O -Wall -Wno-unknown-pragmas -DRTS_VXWORKS -DTOOL=gnu -DTOOL_FAMILY=gnu -DPtrIntType=long -DCPU=PPC604 -DNDEBUG -c -Wp,-MD
	Static or Dynamic Debug	-mcpu=604 -mstrict-align -fno-builtin -ansi -mlongcall -mno-implicit-fp -D_WRS_KERNEL -D_PROTOTYPE_5_0 -g -DVXWORKS_MAJOR_VERSION=6 -DVXWORKS_MINOR_VERSION=5 -O -Wall -Wno-unknown-pragmas -DRTS_VXWORKS -DTOOL=gnu -DTOOL_FAMILY=gnu -DPtrIntType=long -DCPU=PPC604 -c -Wp,-MD
ppc604Vx6.5gcc3.4.4_rtp	Static Release	-mhard-float -mstrict-align -ansi -mregnames -mlongcall -mrtp -D_PROTOTYPE_5_0 -DVXWORKS_MAJOR_VERSION=6 -DVXWORKS_MINOR_VERSION=5 -O -Wall -Wno-unknown-pragmas -DRTS_VXWORKS -DTOOL=gnu -DTOOL_FAMILY=gnu -DPtrIntType=long -DCPU=PPC32 -DNDEBUG -c -Wp,-MD
	Static Debug	-mhard-float -mstrict-align -ansi -mregnames -mlongcall -mrtp -D_PROTOTYPE_5_0 -g -DVXWORKS_MAJOR_VERSION=6 -DVXWORKS_MINOR_VERSION=5 -O -Wall -Wno-unknown-pragmas -DRTS_VXWORKS -DTOOL=gnu -DTOOL_FAMILY=gnu -DPtrIntType=long -DCPU=PPC32 -c -Wp,-MD

Table 11.4 Library-Creation Details for All VxWorks Architectures

RTI Architecture	Library Format	Compiler Flags Used by RTI
ppc604Vx6.6gcc4.1.2	Static or Dynamic Release	-mcpu=604 -fno-builtin -mlongcall -DTOOL=gnu -mstrict-align -ansi -D_WRS_KERNEL -D_PROTOTYPE_5_0 -DVXWORKS_MAJOR_VERSION=6 -DVXWORKS_MINOR_VERSION=6 -O -Wall -Wno-unknown-pragmas -DRTS_VXWORKS -DTOOL=gnu -DTOOL_FAMILY=gnu -DPtrIntType=long -DCPU=PPC604 -DNDEBUG -c -Wp,-MD
	Static or Dynamic Debug	-mcpu=604 -fno-builtin -mlongcall -DTOOL=gnu -mstrict-align -ansi -D_WRS_KERNEL -D_PROTOTYPE_5_0 -g -DVXWORKS_MAJOR_VERSION=6 -DVXWORKS_MINOR_VERSION=6 -O -Wall -Wno-unknown-pragmas -DRTS_VXWORKS -DTOOL=gnu -DTOOL_FAMILY=gnu -DPtrIntType=long -DCPU=PPC604 -c -Wp,-MD
ppc604Vx6.6gcc4.1.2_rtp	Static Release	-mhard-float -mlongcall -mregnames -mstrict-align -ansi -mrtp -D_PROTOTYPE_5_0 -DVXWORKS_MAJOR_VERSION=6 -DVXWORKS_MINOR_VERSION=6 -O -Wall -Wno-unknown-pragmas -DRTS_VXWORKS -DTOOL=gnu -DTOOL_FAMILY=gnu -DPtrIntType=long -DCPU=PPC32 -DNDEBUG -c -Wp,-MD
	Static Debug	-mhard-float -mlongcall -mregnames -mstrict-align -ansi -mrtp -fPIC -shared -D_PROTOTYPE_5_0 -g -DVXWORKS_MAJOR_VERSION=6 -DVXWORKS_MINOR_VERSION=6 -O -Wall -Wno-unknown-pragmas -DRTS_VXWORKS -DTOOL=gnu -DTOOL_FAMILY=gnu -DPtrIntType=long -DCPU=PPC32 -c -Wp,-MD
ppc604Vx6.7gcc4.1.2	Static or Dynamic Release	-mcpu=604 -fno-builtin -mlongcall -DTOOL=gnu -mstrict-align -ansi -D_WRS_KERNEL -D_PROTOTYPE_5_0 -DVXWORKS_MAJOR_VERSION=6 -DVXWORKS_MINOR_VERSION=7 -O -Wall -Wno-unknown-pragmas -DRTS_VXWORKS -DTOOL=gnu -DTOOL_FAMILY=gnu -DPtrIntType=long -DCPU=PPC604 -DNDEBUG -c -Wp,-MD
	Static or Dynamic Debug	-mcpu=604 -fno-builtin -mlongcall -DTOOL=gnu -mstrict-align -ansi -D_WRS_KERNEL -D_PROTOTYPE_5_0 -g -DVXWORKS_MAJOR_VERSION=6 -DVXWORKS_MINOR_VERSION=7 -O -Wall -Wno-unknown-pragmas -DRTS_VXWORKS -DTOOL=gnu -DTOOL_FAMILY=gnu -DPtrIntType=long -DCPU=PPC604 -c -Wp,-MD
ppc604Vx6.7gcc4.1.2_rtp, ppc604Vx6.7gcc4.1.2_smp	Static Release	-mhard-float -mlongcall -mregnames -mstrict-align -ansi -mrtp -D_PROTOTYPE_5_0 -DVXWORKS_MAJOR_VERSION=6 -DVXWORKS_MINOR_VERSION=7 -O -Wall -Wno-unknown-pragmas -DRTS_VXWORKS -DTOOL=gnu -DTOOL_FAMILY=gnu -DPtrIntType=long -DCPU=PPC32 -DNDEBUG -c -Wp,-MD
	Static Debug	-mhard-float -mlongcall -mregnames -mstrict-align -ansi -mrtp -fPIC -shared -D_PROTOTYPE_5_0 -g -DVXWORKS_MAJOR_VERSION=6 -DVXWORKS_MINOR_VERSION=7 -O -Wall -Wno-unknown-pragmas -DRTS_VXWORKS -DTOOL=gnu -DTOOL_FAMILY=gnu -DPtrIntType=long -DCPU=PPC32 -c -Wp,-MD

Table 11.4 Library-Creation Details for All VxWorks Architectures

RTI Architecture	Library Format	Compiler Flags Used by RTI
ppc604Vx6.8gcc4.1.2	Static or Dynamic Release	ccppc -m32 -mstrict-align -ansi -fno-builtin -mlongcall -DCPU=PPC32 -DTOOL_FAMILY=gnu -DTOOL=gnu -D_WRS_KERNEL -D__PROTOTYPE_5_0 -DVXWORKS_MAJOR_VERSION=6 -DVXWORKS_MINOR_VERSION=8 -O2 -fno-strict-aliasing -Wall -Wno-unknown-pragmas -DRTS_VXWORKS -DPtrIntType=long -DCSREAL_IS_FLOAT -DNDEBUG -Wp,-MD
	Static or Dynamic Debug	ccppc -m32 -mstrict-align -ansi -fno-builtin -mlongcall -DCPU=PPC32 -DTOOL_FAMILY=gnu -DTOOL=gnu -D_WRS_KERNEL -D__PROTOTYPE_5_0 -g -DVXWORKS_MAJOR_VERSION=6 -DVXWORKS_MINOR_VERSION=8 -Wall -Wno-unknown-pragmas -DRTS_VXWORKS -DPtrIntType=long -DCSREAL_IS_FLOAT -Wp,-MD
ppc604Vx6.8gcc4.1.2_rtp	Static or Dynamic Release	ccppc -m32 -mhard-float -mstrict-align -mregnames -ansi -mlongcall -DCPU=PPC32 -DTOOL_FAMILY=gnu -DTOOL=gnu -mrtp -D__PROTOTYPE_5_0 -O2 -fno-strict-aliasing -Wall -Wno-unknown-pragmas -DRTS_VXWORKS -DVXWORKS_MAJOR_VERSION=6 -DVXWORKS_MINOR_VERSION=8 -DPtrIntType=long -DCSREAL_IS_FLOAT -DNDEBUG -Wp,-MD
	Static or Dynamic Debug	ccppc -m32 -mhard-float -mstrict-align -mregnames -ansi -mlongcall -DCPU=PPC32 -DTOOL_FAMILY=gnu -DTOOL=gnu-mrtp -D__PROTOTYPE_5_0 -g -Wall -Wno-unknown-pragmas -DRTS_VXWORKS -DVXWORKS_MAJOR_VERSION=6 -DVXWORKS_MINOR_VERSION=8 -DPtrIntType=long -DCSREAL_IS_FLOAT -Wp,-MD
ppc604Vx6.9gcc4.3.3	Static Release	ccppc -m32 -mstrict-align -ansi -fno-builtin -mlongcall -DCPU=PPC32 -DTOOL_FAMILY=gnu -DTOOL=gnu -D_WRS_KERNEL -D__PROTOTYPE_5_0 -DVXWORKS_MAJOR_VERSION=6 -DVXWORKS_MINOR_VERSION=9 -O2 -fno-strict-aliasing -Wall -Wno-unknown-pragmas -DRTS_VXWORKS -DPtrIntType=long -DCSREAL_IS_FLOAT -DNDEBUG -Wp,-MD
	Static Debug	ccppc -m32 -mstrict-align -ansi -fno-builtin -mlongcall -DCPU=PPC32 -DTOOL_FAMILY=gnu -DTOOL=gnu -D_WRS_KERNEL -D__PROTOTYPE_5_0 -g -DVXWORKS_MAJOR_VERSION=6 -DVXWORKS_MINOR_VERSION=9 -Wall -Wno-unknown-pragmas -DRTS_VXWORKS -DPtrIntType=long -DCSREAL_IS_FLOAT -Wp,-MD
ppc604Vx6.9gcc4.3.3_rtp	Static Release	ccppc -mhard-float -mstrict-align -m32 -mregnames -ansi -mlongcall -DCPU=PPC32 -DTOOL_FAMILY=gnu -DTOOL=gnu -mrtp -D__PROTOTYPE_5_0 -O2 -fno-strict-aliasing -Wall -Wno-unknown-pragmas -DRTS_VXWORKS -DVXWORKS_MAJOR_VERSION=6 -DVXWORKS_MINOR_VERSION=9 -DPtrIntType=long -DCSREAL_IS_FLOAT -DNDEBUG -Wp,-MD
	Static Debug	ccppc -mhard-float -mstrict-align -m32 -mregnames -ansi -mlongcall -DCPU=PPC32 -DTOOL_FAMILY=gnu -DTOOL=gnu -mrtp -D__PROTOTYPE_5_0 -g -Wall -Wno-unknown-pragmas -DRTS_VXWORKS -DVXWORKS_MAJOR_VERSION=6 -DVXWORKS_MINOR_VERSION=9 -DPtrIntType=long -DCSREAL_IS_FLOAT -Wp,-MD

Table 11.4 Library-Creation Details for All VxWorks Architectures

RTI Architecture	Library Format	Compiler Flags Used by RTI
ppce500v2Vx6.9.4gcc4.3.3	Static or Dynamic Release	ccppc -m32 -mstrict-align -ansi -fno-builtin -mlongcall -DCPU=PPC32 -DTOOL_ FAMILY=gnu -DTOOL=e500v2gnu -te500v2 -mcpu=8548 -mfloat-gprs=double -mspe=yes -mabi=spe -D_WRS_KERNEL -D__PROTOTYPE_5_0 -DVXWORKS_MAJOR_ VERSION=6 -DVXWORKS_MINOR_VERSION=9 -O2 -fno-strict-aliasing -Wall -Wno-unknown-pragmas -DRTS_VXWORKS -DPtrIntType=long -DCSREAL_IS_FLOAT -DNDEBUG -Wp,-MD
	Static or Dynamic Debug	ccppc -m32 -mstrict-align -ansi -fno-builtin -mlongcall -DCPU=PPC32 -DTOOL_ FAMILY=gnu -DTOOL=e500v2gnu -te500v2 -mcpu=8548 -mfloat-gprs=double -mspe=yes -mabi=spe -D_WRS_KERNEL -D__PROTOTYPE_5_0 -g -DVXWORKS_MAJOR_ VERSION=6 -DVXWORKS_MINOR_VERSION=9 -Wall -Wno-unknown-pragmas -DRTS_VXWORKS -DPtrIntType=long -DCSREAL_IS_FLOAT -Wp,-MD
ppce500v2Vx6.9.4gcc4.3.3_ rtp	Static or Dynamic Release	ccppc -mstrict-align -m32 -mregnames -ansi -mlongcall -DCPU=PPC32 -DTOOL_ FAMILY=gnu -DTOOL=gnu -te500v2 -mcpu=8548 -mfloat-gprs=double -mspe=yes -mabi=spe -mrtp -D__PROTOTYPE_5_0 -O2 -fno-strict-aliasing -Wall -Wno-unknown-pragmas -DRTS_VXWORKS -DVXWORKS_MAJOR_VERSION=6 -DVXWORKS_ MINOR_VERSION=9 -DPtrIntType=long -DCSREAL_IS_FLOAT -DNDEBUG -Wp,-MD
	Static or Dynamic Debug	ccppc -mstrict-align -m32 -mregnames -ansi -mlongcall -DCPU=PPC32 -DTOOL_ FAMILY=gnu -DTOOL=gnu -te500v2 -mcpu=8548 -mfloat-gprs=double -mspe=yes -mabi=spe -mrtp -D__PROTOTYPE_5_0 -g -Wall -Wno-unknown-pragmas -DRTS_ VXWORKS -DVXWORKS_MAJOR_VERSION=6 -DVXWORKS_MINOR_VERSION=9 -DPtrIntType=long -DCSREAL_IS_FLOAT -Wp,-MD
sbc8641Vx653-2.3gcc3.3.2	Static or Dynamic Release	-DTOOL_FAMILY=gnu -DTOOL=gnu -mlongcall -Wall -G 0 -fno-builtin -mlongcall -D_ WRS_KERNEL -D__PROTOTYPE_5_0 -DVXWORKS_MAJOR_VERSION=5 -DVXWORKS_MINOR_VERSION=5 -O -Wno-unknown-pragmas -DRTS_VXWORKS -DPtrIntType=long -DCSREAL_IS_FLOAT -DCPU=PPC604 -DNDEBUG -c -Wp,-MD
	Static or Dynamic Debug	-DTOOL_FAMILY=gnu -DTOOL=gnu -mlongcall -Wall -G 0 -fno-builtin -mlongcall -D_ WRS_KERNEL -D__PROTOTYPE_5_0 -g -DVXWORKS_MAJOR_VERSION=5 -DVXWORKS_MINOR_VERSION=5 -Wall -Wno-unknown-pragmas -DRTS_VXWORKS -DPtrIntType=long -DCSREAL_IS_FLOAT -DCPU=PPC604 -c -Wp,-MD
simpcVx653-2.3gcc3.3.2	Static or Dynamic Release	-DTOOL_FAMILY=gnu -DTOOL=gnu -DCPU=SIMNT -Wall -nostdlib -fno-defer-pop -fno-builtin -mcpu=pentium -D_WRS_KERNEL -D__PROTOTYPE_5_0 -DVXWORKS_ MAJOR_VERSION=5 -DVXWORKS_MINOR_VERSION=5 -O -Wno-unknown-pragmas -DRTS_VXWORKS -DPtrIntType=long -DCSREAL_IS_FLOAT -DCPU=SIMNT -DNDEBUG -c -Wp,-MD
	Static or Dynamic Debug	-DTOOL_FAMILY=gnu -DTOOL=gnu -DCPU=SIMNT -Wall -nostdlib -fno-defer-pop -fno-builtin -mcpu=pentium -D_WRS_KERNEL -D__PROTOTYPE_5_0 -g -DVXWORKS_ MAJOR_VERSION=5 -DVXWORKS_MINOR_VERSION=5 -Wno-unknown-pragmas -DRTS_VXWORKS -DPtrIntType=long -DCSREAL_IS_FLOAT -DCPU=SIMNT -c -Wp,-MD

11.1 Required Makefile Change for VxWorks 7.0 Platforms

For VxWorks 7.0 platforms only: After you run *rtiddsgen*, edit the generated makefile to specify which Vxworks Source Build (VSB) you want to use. In the generated makefile, find this line and change it to match your VSB directory:

```
32 VSB_DIR = # Specify your VSB directory here.
```

Note: RTI uses VSB 2.1.2.2 to build the Connext DDS libraries for VxWorks 7.0.

11.2 Request-Reply Communication Pattern

The Connext DDS Professional, Research, Evaluation, and Basic packages include support for the Request-Reply Communication Pattern, for all platforms in Table 11.1 Supported VxWorks Target Platforms and all programming languages, except as noted below.

When using C++, the following platforms do not support the Request-Reply Communication Pattern:

- ppc603Vx5.5gcc
- ppc604Vx5.5gcc

When using a Connext DDS dynamic library for C++ Request-Reply for kernel-mode, you need to perform an extra host processing step called *munching* and apply it to any application that is linking against the C++ Request-Reply library.

In VxWorks kernel-mode, before a C++ module can be downloaded to the VxWorks kernel, it must undergo an additional host processing step, known as *munching*. This step is necessary for properly initialization of static objects and to ensure that the C++ run-time support calls the correct constructor/destructors in the correct order for all static objects.

If you need to use the C++ Request-Reply API for kernel-mode with dynamically linked libraries, you need to *munch* your application and link or load the Connext DDS library for C++ request/reply, in addition to the standard Connext DDS libraries for core, C, and C++.

RTI provides pre-munched Connext DDS dynamic libraries for C++ Request-Reply with the extension "**.lo**". For example, if you plan to load your application at run-time for kernel-mode and your application uses the Request-Reply API for C++ with dynamic libraries, assuming you want to use non-debug libraries, you need to first load the **libnddscore.so** library, then **libnddsc.so**, then **libnddscpp.so**, and finally **librticonnextmsgcpp.lo**. Once all these libraries are loaded, you can load your munched C++ application.

The following table shows the libraries for which RTI has performed the munching process.

Table 11.5 Pre-Munched Kernel-mode C++ Request-Reply Dynamic Libraries

Library	Description
librticonnextmsgcpp.lo	Munched Release C++ Request-Reply library
librticonnextmsgcppd.lo	Munched Debug C++ Request-Reply library

11.3 Increasing the Stack Size

Connext DDS applications may require more than the default stack size on VxWorks.

To prevent stack overrun, you can create/enable the *DomainParticipant* in a thread with a larger stack, or increase the default stack size of the shell task by recompiling the kernel. For more information, please see the Solutions on the RTI Customer Portal, accessible from https://support.rti.com/.

11.4 Libraries for RTP Mode on VxWorks 6.3 and Higher Systems

Dynamic libraries are *not* available for VxWorks 6.3 and higher systems with Real Time Processes (RTP mode) on PowerPC (PPC) CPUs. This is due to a platform limitation in VxWorks PPC platforms that puts an upper bound on the size of the Global Offset Table (GOT) for any single library, which limits how many symbols the library can export. Some Connext DDS libraries (in particular, libnddsc) export a number of symbols that exceed this upper bound.

Dynamic libraries *are* available for VxWorks 6.3 and higher systems with RTP mode on Pentimum CPUs.

11.5 Requirement for Restarting Applications

When restarting a VxWorks application, you may need to change the 'appId' value. In general, this is only required if you still have other Connext DDS applications running on other systems that were talking to the restarted application. If all the Connext DDS applications are restarted, there should be no problem.

This section explains why this is necessary and how to change the appId.

All Connext DDS applications must have a unique GUID (globally unique ID). This GUID is composed of a hostId and an appId. RTI implements unique appIds by using the process ID of the application. On VxWorks systems, an application's process ID will often be the same across reboots. This may cause logged errors during the discovery process, or discovery may not complete successfully for the restarted application.

The workaround is to manually provide a unique appId each time the application starts. The appId is stored in the *DomainParticipant's* WireProtocol QosPolicy. There are two general approaches to providing a unique appId. The first approach is to save the appId in NVRAM or the file system, and then increment the appId across reboots. The second approach is to base the appId on something that is likely to be different across reboots, such as a time-based register.

11.6 Support for Modern C++ API

Connext DDS provides two different C++ APIs, which we refer to as the "Traditional C++" and "Modern C++" APIs. The Modern C++ API is available for all VxWorks platforms except VxWorks 5.5 and VxWorks 653.

For more information on Modern C++, see "Traditional vs. Modern C++" in the User's Manual.

11.7 Multicast Support

Multicast is supported on VxWorks 5.x - 7.x and VxWorks 653 on sbc8641d CPU platforms. It is configured out of the box. That is, the default value for the initial peers list (NDDS_DISCOVERY_PEERS) includes a multicast address. See the API Reference HTML documentation for more information.

Multicast is *not* supported on the following platforms:

- VxWorks 653 on the SIMPC CPU

Known Defects:

- If you have a Wind River account, you can find more information about defect WIND00418701 here: https://support.windriver.com/olsPortal/faces/maintenance/defectDetails.jspx?defectId=WIND00418701.

  This issue has been fixed in VxWorks 6.9.3.2. If you need a patch for your version of VxWorks, or for more information about this issue, please contact Wind River.

- There is a known defect when using VxWorks 6.9.3.2 in a multicast scenario. If you have a Wind River account, you can find more information about defect VXW6-8077 here: https://support.windriver.com/olsPortal/faces/maintenance/defectDetails.jspx?defectId=VXW6-80771&_adf.ctrl-state=crbf0uqpa_4

  If you are using VxWorks 6.9.3.2 and want to use multicast, please contact Wind River to get an official patch to fix this issue.

11.8 Supported Transports

Shared memory: Shared memory is supported and enabled by default on all VxWorks 6.x and higher architectures. It is not supported on VxWorks 5.x and VxWorks 653 platforms. See also:

- Shared-Memory Communication between Applications Running in Kernel Mode and RTP Requires Explicitly Set Participant ID (Section 11.8.1 on the next page)

- How To Run Connext DDS Libraries in Kernels Built without Shared Memory (Section 11.8.2 below)

UDPv4: Supported and enabled by default.

UDPv6: Supported on VxWorks 6.7 and higher architectures except as noted below.
No Traffic Class support.

TCP/IPv4: Not supported.

11.8.1 Shared-Memory Communication between Applications Running in Kernel Mode and RTP Requires Explicitly Set Participant ID

By default, applications using the auto-generated Participant ID (-1) cannot communicate between user space and kernel space on the same host via SHMEM. The root cause is that the participants use the same participant ID. Therefore the workaround for this issue is to explicitly provide a participant ID when creating the *DomainParticipants*. The participant ID is set in the *DomainParticipant's* WireProtocol QoS policy.

11.8.2 How To Run Connext DDS Libraries in Kernels Built without Shared Memory

Since Connext DDS libraries support shared memory as a built-in transport, building a kernel without shared-memory support will cause loading or linking errors, depending on whether the Connext DDS libraries are loaded after boot, or linked at kernel build time.

The most straightforward way to fix these errors is to include shared-memory support in the kernel (INCLUDE_SHARED_DATA in the kernel build parameters).

However, in some versions of VxWorks, it is not possible to include shared-memory support without also including RTP support. If you are unwilling or unable to include shared-memory support in your configuration, you will need to do the following:

1. Add the component INCLUDE_POSIX_SEM
2. Define stubs that return failure for the missing symbols **sdOpen** and **sdUnmap** as described below:
   - For **sdOpen**, we recommend providing an implementation that returns NULL, and sets errno to ENOSYS. For the function prototype, refer to the file **sdLib.h** in the VxWorks distribution.
   - For **sdUnmap**, we recommend providing an implementation that returns ERROR and sets errno to ENOSYS. For the function prototype, refer to the file **sdLibCommon.h** in the VxWorks distribution.

In addition to providing the symbol stubs for **sdOpen** and **sdUnmap**, we also recommend disabling the SHMEM transport by using the **transport_builtin** mask in the QoS configuration.

11.9 Monotonic Clock Support

The monotonic clock (described in "Clock Selection" in the User's Manual) is supported on VxWorks 6.3 and higher platforms. This feature is not supported on VxWorks 653 2.3 platforms.

11.10 Thread Configuration

Table 11.6 Thread Setting for VxWorks Platforms (Applies to Kernel Tasks or Real-Time Process Threads) lists the thread settings for VxWorks platforms.

Table 11.7 Thread-Priority Definitions for VxWorks Platforms and Table 11.8 Thread Kinds for VxWorks Platforms list the thread-priority definitions and thread kinds, respectively.

11.10.1 Support for Controlling CPU Core Affinity for RTI Threads

Support for controlling CPU core affinity (described in "Controlling CPU Core Affinity" in the User's Manual) is not available for VxWorks platforms.

Table 11.6 Thread Setting for VxWorks Platforms (Applies to Kernel Tasks or Real-Time Process Threads)

Applicable Thread	DDS_ ThreadSettings_t	Platform-Specific Setting
Asynchronous Publisher, Asynchronous flushing thread	mask	OS default thread type
	priority	100
	stack_size	30 * 1024
	cpu_list	CPU core affinity not supported
	cpu_rotation	CPU core affinity not supported
Database thread	mask	DDS_THREAD_SETTINGS_STDIO
	priority	120
	stack_size	30 * 1024
	cpu_list	CPU core affinity not supported
	cpu_rotation	CPU core affinity not supported

Table 11.6 Thread Setting for VxWorks Platforms (Applies to Kernel Tasks or Real-Time Process Threads)

Applicable Thread	DDS_ ThreadSettings_t	Platform-Specific Setting
Event thread	mask	DDS_THREAD_SETTINGS_STDIO | DDS_THREAD_SETTINGS_ FLOATING_POINT
	priority	110
	stack_size	4 * 30 * 1024
	cpu_list	CPU core affinity not supported
	cpu_rotation	CPU core affinity not supported
ReceiverPool threads	mask	DDS_THREAD_SETTINGS_STDIO | DDS_THREAD_SETTINGS_ FLOATING_POINT
	priority	71
	stack_size	4 * 30 * 1024
	cpu_list	CPU core affinity not supported
	cpu_rotation	CPU core affinity not supported

Table 11.7 Thread-Priority Definitions for VxWorks Platforms

Thread-Priority Definition	Operating-System Priority
THREAD_PRIORITY_DEFAULT	100
THREAD_PRIORITY_HIGH	68
THREAD_PRIORITY_ABOVE_NORMAL	71
THREAD_PRIORITY_NORMAL	100
THREAD_PRIORITY_BELOW_NORMAL	110
THREAD_PRIORITY_LOW	120

Table 11.8 Thread Kinds for VxWorks Platforms

Thread Kinds	Operating-System Configuration[a]
DDS_THREAD_SETTINGS_FLOATING_POINT	Uses VX_FP_TASK when calling taskSpawn()
DDS_THREAD_SETTINGS_STDIO	Uses VX_STDIO when calling taskSpawn() (Kernel mode only)
DDS_THREAD_SETTINGS_REALTIME_PRIORITY	Configures the schedule policy to SCHED_FIFO.
DDS_THREAD_SETTINGS_PRIORITY_ENFORCE	N/A

11.11 Durable Writer History and Durable Reader State Features

The Durable Writer History and Durable Reader State features are not supported on VxWorks platforms.

11.12 Libraries Required for Using Distributed Logger

RTI Distributed Logger is only supported on these VxWorks architectures:

- VxWorks 6.7:
  - ppc604Vx6.7gcc4.1.2
  - ppc604Vx6.7gcc4.1.2_rtp
- VxWorks 6.8:
  - ppc604Vx6.8gcc4.1.2
  - ppc604Vx6.8gcc4.1.2_rtp
- VxWorks 6.9.4:
  - ppce500v2Vx6.9.4gcc4.3.3
  - ppce500v2Vx6.9.4gcc4.3.3_rtp
- Works 7.0
  - pentiumVx7.0gcc4.3.3
  - pentiumVx7.0gcc4.3.3_rtp

Table 11.9 Additional Libraries for using RTI Distributed Logger lists the additional libraries you will need in order to use *Distributed Logger*.

[a]See VxWorks manuals for additional information.

Table 11.9 Additional Libraries for using RTI Distributed Logger

Language	Static		Dynamic	
	Release	Debug	Release	Debug
C	librtidlcz.a	librtidlczd.a	librtidlc.so	librtidcd.so
C++ (Traditional API)	librtidlcz.a librtidlcppz.a	librtidlczd.a librtidlcppzd.a	librtidlc.so librtidlcpp.so	librtidlcd.so librtidlcppd.so

11.13 Libraries Required for Using Monitoring

Make sure you are consistent in your use of static, dynamic, debug and release versions of the libraries. For example, if your Connext DDS application is linked with the static release version of the Connext DDS libraries, you will need to also use the static release version of the monitoring library. Do not link both static and dynamic libraries. Similarly, do not mix release and debug libraries.

If you are statically linking your application with DDS libraries and you want to add monitoring to your application, you will also need to statically link the monitoring library. The library cannot be loaded dynamically strictly through the QoS profile because it also depends on DDS to publish its data. Therefore, it depends on DDS; the DDS functionality would cause duplicate symbols to be found resulting, in the termination of the process.

Note: If you plan to use *static* libraries, the RTI library from Table 11.10 Additional Libraries for Using Monitoring must appear *first* in the list of libraries to be linked.

Table 11.10 Additional Libraries for Using Monitoring

Static Release	Static Debug	Dynamic Release	Dynamic Debug
librtimonitoringz.a	librtimonitoringzd.a	librtimonitoring.so[a]	librtimonitoringd.so[b]

11.14 Increasing the Receive Socket Buffer Size

For Connext DDS applications running on VxWorks 6.7 or higher systems and using UDPv4, we recommend setting the property **dds.transport.UDPv4.builtin.recv_socket_buffer_size** to a value of 128000 or higher. This recommendation is due to Wind River's usage of extra receive socket buffer space to correct Wind River defect number WIND00135312.

---

[a]Dynamic libraries are not supported for VxWorks platforms on PPC CPUs using RTP mode.

[b]Dynamic libraries are not supported for VxWorks platforms on PPC CPUs using RTP mode.

Table 11.11 Required Kernel Components for sbc8641Vx653-2.3gcc3.3.2[b]

INCLUDE_ARINC_SCHEDULER_INIT	INCLUDE_NETINET_IF_SUBR
INCLUDE_ARP_API	INCLUDE_NETINET_IGMP
INCLUDE_AUXCLK	INCLUDE_NETINET_IN
INCLUDE_BOOT_LINE	INCLUDE_NETINET_IN_CKSUM
INCLUDE_BOOT_LINE_INIT	INCLUDE_NETINET_IN_PCB
INCLUDE_BSD_SOCKET	INCLUDE_NETINET_IN_PROTO
INCLUDE_BSP_MODULES	INCLUDE_NETINET_IP_ICMP
INCLUDE_BSP_VXWORKS	INCLUDE_NETINET_IP_INPUT
INCLUDE_BYTENVRAM	INCLUDE_NETINET_IP_OUTPUT
INCLUDE_DEBUG_CORE	INCLUDE_NETINET_RADIX
INCLUDE_DEBUG_UTIL	INCLUDE_NETINET_RAW_IP
INCLUDE_END	INCLUDE_NETINET_ROUTE
INCLUDE_END_BOOT	INCLUDE_NETINET_SYS_SOCKET
INCLUDE_EXC_SHOW_INIT	INCLUDE_NETINET_UDP_USRREQ
INCLUDE_FLASHMEM	INCLUDE_NETINET_UIPC_DOM
INCLUDE_FTP	INCLUDE_NETINET_UIPC_MBUF
INCLUDE_HOST_TBL	INCLUDE_NETINET_UIPC_SOCK
INCLUDE_ICMP	INCLUDE_NETINET_UIPC_SOCK2
INCLUDE_IGMP	INCLUDE_NETINET_UNIXLIB
INCLUDE_IO_EXTRA_INIT	INCLUDE_NETMASK_GET
INCLUDE_IO_SYSTEM_INIT	INCLUDE_NETWORK
INCLUDE_IP	INCLUDE_NETWRS_ETHERMULTILIB
INCLUDE_KERNEL_BASIC	INCLUDE_NETWRS_IFLIB

[b]Install partition_socket_driver_v1.3. Follow instructions from Wind River for the installation.

Table 11.11 Required Kernel Components for sbc8641Vx653-2.3gcc3.3.2[b]

INCLUDE_KERNEL_BASIC_INIT	INCLUDE_NETWRS_INETLIB
INCLUDE_KERNEL_BASIC_INIT2	INCLUDE_NETWRS_NETBUFLIB
INCLUDE_KERNEL_CORE	INCLUDE_NETWRS_REMLIB
INCLUDE_KERNEL_FULL	INCLUDE_NETWRS_ROUTELIB
INCLUDE_KERNEL_NORMAL_MODE	INCLUDE_NETWRS_XDR
INCLUDE_KERNEL_SHOW	INCLUDE_NV_RAM
INCLUDE_KERNEL_UTIL	INCLUDE_PARTITION_INIT
INCLUDE_LOADER	INCLUDE_POST_KERNEL_CORE_INIT
INCLUDE_LOADER_EXTRA	INCLUDE_POST_KERNEL_CORE_INIT2
INCLUDE_LOOPBACK	INCLUDE_PPCDECTIMER
INCLUDE_MIILIB	INCLUDE_PRE_KERNEL_CORE_INIT
INCLUDE_MMU_BASIC	INCLUDE_SERIAL
INCLUDE_MOTTSECEND	INCLUDE_SHELL
INCLUDE_MUX	INCLUDE_SHELL_VI_MODE
INCLUDE_NET_DRV	INCLUDE_SOCKET_DEV
INCLUDE_NET_HOST_SETUP	INCLUDE_SYM_TBL_INIT
INCLUDE_NET_INIT	INCLUDE_SYSCLK
INCLUDE_NET_LIB	INCLUDE_SYSTEM_START_INIT
INCLUDE_NET_RANDOM	INCLUDE_TCP
INCLUDE_NET_REM_IO	INCLUDE_TFTP_CLIENT
INCLUDE_NET_SETUP	INCLUDE_TIME_MONITOR_INIT
INCLUDE_NET_SYM_TBL	INCLUDE_UDP
INCLUDE_NET_TASK	INCLUDE_USER_APPL

[b]Install partition_socket_driver_v1.3. Follow instructions from Wind River for the installation.

Table 11.11 Required Kernel Components for sbc8641Vx653-2.3gcc3.3.2[b]

INCLUDE_NETDEV_CONFIG	INCLUDE_USR_DEVSPLIT
INCLUDE_NETDEV_NAMEGET	INCLUDE_USR_FS_UTILS
INCLUDE_NETINET_IF	INCLUDE_WDB
INCLUDE_NETINET_IF_ETHER	INCLUDE_WDB_COMM_END[a]

Table 11.12 Required Kernel Components for simpcVx653-2.3gcc3.3.2[d]

INCLUDE_ARINC_SCHEDULER_INIT	INCLUDE_NETINET_IN_PCB
INCLUDE_ARP_API	INCLUDE_NETINET_IN_PROTO
INCLUDE_BOOT_LINE	INCLUDE_NETINET_IP_ICMP
INCLUDE_BOOT_LINE_INIT	INCLUDE_NETINET_IP_INPUT
INCLUDE_BSD_SOCKET	INCLUDE_NETINET_IP_OUTPUT
INCLUDE_BSP_MODULES	INCLUDE_NETINET_RADIX
INCLUDE_BSP_VXWORKS	INCLUDE_NETINET_RAW_IP
INCLUDE_DEBUG_CORE	INCLUDE_NETINET_ROUTE
INCLUDE_DEBUG_UTIL	INCLUDE_NETINET_SYS_SOCKET
INCLUDE_END	INCLUDE_NETINET_UDP_USRREQ
INCLUDE_END_BOOT	INCLUDE_NETINET_UIPC_DOM
INCLUDE_FTP	INCLUDE_NETINET_UIPC_MBUF
INCLUDE_HOST_TBL	INCLUDE_NETINET_UIPC_SOCK
INCLUDE_ICMP	INCLUDE_NETINET_UIPC_SOCK2

[a]SELECT_WDB_COMM_TYPE can only have one type at a time. In order to add INCLUDE_WDB_COMM_END, you should remove INCLUDE_WDB_COMM_PIPE.

[b]Install partition_socket_driver_v1.3. Follow instructions from Wind River for the installation.

[d]Install partition_socket_driver_v1.3. Follow instructions from Wind River for the installation.

Table 11.12 Required Kernel Components for simpcVx653-2.3gcc3.3.2[b]

INCLUDE_IGMP	INCLUDE_NETINET_UNIXLIB
INCLUDE_IO_EXTRA_INIT	INCLUDE_NETMASK_GET
INCLUDE_IO_SYSTEM_INIT	INCLUDE_NETWORK
INCLUDE_IP	INCLUDE_NETWRS_ETHERMULTILIB
INCLUDE_KERNEL_BASIC	INCLUDE_NETWRS_IFLIB
INCLUDE_KERNEL_BASIC_INIT	INCLUDE_NETWRS_INETLIB
INCLUDE_KERNEL_BASIC_INIT2	INCLUDE_NETWRS_NETBUFLIB
INCLUDE_KERNEL_CORE	INCLUDE_NETWRS_REMLIB
INCLUDE_KERNEL_FULL	INCLUDE_NETWRS_ROUTELIB
INCLUDE_KERNEL_NORMAL_MODE	INCLUDE_NETWRS_XDR
INCLUDE_LOOPBACK	INCLUDE_NTEND
INCLUDE_MUX	INCLUDE_NTPASSFS
INCLUDE_NET_DRV	INCLUDE_NULLNVRAM
INCLUDE_NET_HOST_SETUP	INCLUDE_PARTITION_INIT
INCLUDE_NET_INIT	INCLUDE_POST_KERNEL_CORE_INIT
INCLUDE_NET_LIB	INCLUDE_POST_KERNEL_CORE_INIT2
INCLUDE_NET_RANDOM	INCLUDE_PRE_KERNEL_CORE_INIT
INCLUDE_NET_REM_IO	INCLUDE_SIMPCTIMER
INCLUDE_NET_SETUP	INCLUDE_SOCKET_DEV
INCLUDE_NET_TASK	INCLUDE_SYSTEM_START_INIT
INCLUDE_NETDEV_CONFIG	INCLUDE_TCP
INCLUDE_NETDEV_NAMEGET	INCLUDE_TFTP_CLIENT
INCLUDE_NETINET_IF	INCLUDE_TIME_MONITOR_INIT

[b]Install partition_socket_driver_v1.3. Follow instructions from Wind River for the installation.

Table 11.12 Required Kernel Components for simpcVx653-2.3gcc3.3.2[b]

INCLUDE_NETINET_IF_ETHER	INCLUDE_UDP
INCLUDE_NETINET_IF_SUBR	INCLUDE_USER_APPL
INCLUDE_NETINET_IGMP	INCLUDE_WDB
INCLUDE_NETINET_IN	INCLUDE_WDB_COMM_END[a]
INCLUDE_NETINET_IN_CKSUM	INCLUDE_WINSIO

[a]SELECT_WDB_COMM_TYPE can only have one type at a time. In order to add INCLUDE_WDB_COMM_END, you should remove INCLUDE_WDB_COMM_PIPE.

[b]Install partition_socket_driver_v1.3. Follow instructions from Wind River for the installation.

Chapter 12 Windows Platforms

First, see the basic instructions for compiling on Windows systems in the "Building Applications" chapter in the User's Manual.

The following tables provide supplemental information. Table 12.1 Supported Windows Platforms lists the architectures supported on Windows operating systems.

Table 12.1 Supported Windows Platforms

Operating System	CPU	Visual Studio® Version	RTI Architecture Abbreviation	.NET Version[a]	JDK Version
Windows 7	x86	VS 2010 SP1	i86Win32VS2013	4.0	1.7 or 1.8
	x64	VS 2010 SP1	x64Win64VS2010	4.0	
Windows 8	x86	VS 2012 Update 4	i86Win32VS2012	4.5	
		VS 2013 Update 4	i86Win32VS2013	4.5.1	
	x64	VS 2012 Update 4	x64Win64VS2012	4.5	
		VS 2013 Update 4	x64Win64VS2013	4.5.1	
Windows 8.1	x86	VS 2013 Update 4	i86Win32VS2013	4.5.1	
	x64	VS 2013 Update 4	x64Win64VS2013	4.5.1	
Windows 10	x86	VS 2015 Update 1	i86Win32VS2015	4.6	
	x64	VS 2015 Update 1	x64Win64VS2015	4.6	
Windows 2003	x86	VS 2008 SP1	i86Win32VS2008	2.0	
	x64	VS 2008 SP1	x64Win64VS2008	2.0	
Windows Server 2008 R2	x64	VS 2010 SP1	x64Win64VS2010	4.0	
Windows Server 2012 R2	x64	VS 2012 Update 4	x64Win64VS2012	4.5	
		VS 2013 Update 4	x64Win64VS2013	4.5.1	
		VS 2015 Update 1	x64Win64VS2015	4.6	
Windows Vista	x86	VS 2008 SP1	i86Win32VS2008	2.0	
	x64	VS 2008 SP1	x64Win64VS2008	2.0	

[a]The RTI .NET assemblies are supported for both the C++/CLI and C# languages. The type support code generated by ***rtiddsgen*** is in C++/CLI; compiling the generated type support code requires Microsoft Visual C++. Calling the assembly from C# requires Microsoft Visual C#.

Table 12.1 Supported Windows Platforms

Operating System	CPU	Visual Studio® Version	RTI Architecture Abbreviation	.NET Version[a]	JDK Version
Windows XP Pro SP2 [b] [c]	x86	VS 2008 SP1	i86Win32VS2008	2.0	
	x64	VS 2008 SP1	x64Win64VS2008	2.0	

The compiler flags and the libraries you will need to link into your application are listed in the following tables:

- Windows host platforms: Table 12.2 Building Instructions for Windows Host Architectures
- Windows target platforms: Table 12.3 Building Instructions for Windows Target Architectures

See also:

- Libraries Required for Using Distributed Logger Support (Section 12.13 on page 159)
- Libraries Required for Using RTI Secure WAN Transport APIs (Section 12.15 on page 159)
- Libraries Required for Using RTI TCP Transport APIs (Section 12.16 on page 160)

To use libraries that are *statically* linked into an application, link in all of the libraries listed in one of the rows of these tables. To use *dynamic* link libraries (DLL) on Windows systems, link in all of the libraries listed in one of the 'Dynamic' sections of the appropriate table. When the application executes, it will attempt to dynamically link in the libraries, which are in the directory **$(NDDSHOME)\lib\<*architecture*>** (this directory must be placed on the path before the executable is started).

Windows libraries are provided in formats with and without debugging symbols. Choose the format appropriate for your current work. Do not mix libraries built for different formats.

---

[a]The RTI .NET assemblies are supported for both the C++/CLI and C# languages. The type support code generated by ***rtiddsgen*** is in C++/CLI; compiling the generated type support code requires Microsoft Visual C++. Calling the assembly from C# requires Microsoft Visual C#.

[b]Windows XP: If you are using JDK 5.0 and want to use Intel's HyperThreading technology, use JDK 5.0 Update 6 (build 1.5.0_06), which includes fixes to JNI and HyperThreading. (If you must use Update 5 (build 1.5.0_05), you should disable HyperThreading.)

[c]Windows XP does not support IP_TOS unless registry changes are made. See http://support.microsoft.com/kb/248611, http://www.microsoft.com/technet/technetmag/issues/2007/02/CableGuy/default.aspx.

Table 12.4 Running Instructions for Windows Architectures provides details on the environment variables that must be set at run time for a Windows architecture.

For details on how the libraries were built by RTI, see Table 12.5 Library-Creation Details for Windows Architectures. This information is provided strictly for informational purposes; you do not need to use these parameters to compile your application. You may find this information useful if you are involved in any in-depth debugging.

Table 12.2 Building Instructions for Windows Host Architectures

API	Library Format	RTI Libraries or Jar Files [a] [b] [c]	Required System Libraries	Required Compiler Flags
C	Static Release	nddscz.lib nddscorez.lib rticonnextmsgcz.lib	netapi32.lib advapi32.lib user32.lib ws2_32.lib	/D “RTI_WIN32” /MT
	Static Debug	nddsczd.lib nddscorezd.lib rticonnextmsgczd.lib		/D “RTI_WIN32” /MTd
	Dynamic Release	nddsc.lib nddscore.lib rticonnextmsgc.lib		/D “RTI_WIN32” /D “NDDS_DLL_VARIABLE” /MD
	Dynamic Debug	nddscd.lib nddscored.lib rticonnextmsgcd.lib		/D “RTI_WIN32” /D “NDDS_DLL_VARIABLE” /MDd

[a] Choose nddscpp*.* for the Traditional C++ API or nddscpp2*.* for the Modern C++ API.

[b] The RTI C/C++/Java libraries are in <NDDSHOME>\lib\<architecture>. Jar files are in <NDDSHOME>\lib\java.

[c] Some library names include a [***version***], which depends on your version of .NET.
For .NET 2.0, omit the [***version***].
For other .NET versions, use the digits, such as 451 or 46. See **Table 12.1 Supported Windows Platforms**.for supported .NET versions.

Table 12.2 Building Instructions for Windows Host Architectures

API	Library Format	RTI Libraries or Jar Files [a] [b] [c]	Required System Libraries	Required Compiler Flags
C++ (Traditional and Modern APIs)	Static Release	nddscppz.lib or nddscpp2z.lib nddscz.lib nddscorez.lib rticonnextmsgcppz.lib	netapi32.lib advapi32.lib user32.lib ws2_32.lib	/D “RTI_WIN32” /MT
	Static Debug	nddscppzd.lib or nddscpp2zd.lib nddsczd.lib nddscorezd.lib rticonnextmsgcppzd.lib		/D “RTI_WIN32” /MTd
	Dynamic Release	nddscpp.lib or nddscpp2.lib nddsc.lib nddscore.lib rticonnextmsgcpp.lib		/D “RTI_WIN32” /D “NDDS_DLL_VARIABLE” /MD
	Dynamic Debug	nddscppd.lib or nddscpp2d.lib nddscd.lib nddscored.lib rticonnextmsgcppd.lib		/D “RTI_WIN32” /D “NDDS_DLL_VARIABLE” /MDd

[a] Choose nddscpp*.* for the Traditional C++ API or nddscpp2*.* for the Modern C++ API.

[b] The RTI C/C++/Java libraries are in <NDDSHOME>\lib\<architecture>. Jar files are in <NDDSHOME>\lib\java.

[c] Some library names include a [***version***], which depends on your version of .NET.
For .NET 2.0, omit the [***version***].
For other .NET versions, use the digits, such as 451 or 46. See **Table 12.1 Supported Windows Platforms**.for supported .NET versions.

Table 12.2 Building Instructions for Windows Host Architectures

API	Library Format	RTI Libraries or Jar Files [a] [b] [c]	Required System Libraries	Required Compiler Flags
C++/CLI	Release	nddscpp.lib nddsc.lib nddscore.lib nddsdotnet[*version*].dll rticonnextmsgdotnet [*version*].dll	N/A	/D “RTI_WIN32” /D “NDDS_DLL_ VARIABLE” /MD /D “WIN32_LEAN_ AND_MEAN”
	Debug	nddscppd.lib nddscd.lib nddscored.lib nddsdotnet[*version*]d.dll rticonnextmsgdotnet[*version*] d.dll		/D “RTI_WIN32” /D “NDDS_DLL_ VARIABLE” /MDd /D “WIN32_LEAN_ AND_MEAN”
C#	Release	nddsdotnet[*version*].dll rticonnextmsgdotnet [*version*].dll	N/A	N/A
	Debug	nddsdotnet[*version*]d.dll rticonnextmsgdotnet[*version*] d.dll		
Java	Release	nddsjava.jar rticonnextmsg.jar	N/A	N/A
	Debug	nddsjavad.jar rticonnextmsgd.jar		

[a] Choose nddscpp*.* for the Traditional C++ API or nddscpp2*.* for the Modern C++ API.

[b] The RTI C/C++/Java libraries are in <NDDSHOME>\lib\<architecture>. Jar files are in <NDDSHOME>\lib\java.

[c] Some library names include a [***version***], which depends on your version of .NET.
For .NET 2.0, omit the [***version***].
For other .NET versions, use the digits, such as 451 or 46. See **Table 12.1 Supported Windows Platforms**.for supported .NET versions.

Table 12.3 Building Instructions for Windows Target Architectures

API	Library Format	RTI Libraries or Jar Files [a] [b] [c]	Required System Libraries	Required Compiler Flags
C	Static Release	nddscz.lib nddscorez.lib rticonnextmsgcz.lib	netapi32.lib advapi32.lib user32.lib ws2_32.lib	/Gd /MT /D “WIN32” /D “RTI_WIN32” /D “NDEBUG”
	Static Debug	nddsczd.lib nddscorezd.lib rticonnextmsgczd.lib		/Gd /MTd /D “WIN32” /D “RTI_WIN32”
	Dynamic Release	nddsc.lib nddscore.lib rticonnextmsgc.lib		/Gd /MD /D “WIN32” /D “NDDS_DLL_ VARIABLE” /D “RTI_WIN32” /D “NDEBUG”
	Dynamic Debug	nddscd.lib nddscored.lib rticonnextmsgcd.lib		/Gd /MDd /D “WIN32” /D “NDDS_DLL_ VARIABLE” /D “RTI_WIN32”

[a]The RTI C/C++/Java libraries are in <NDDSHOME>\lib\<architecture>. Jar files are in <NDDSHOME>\lib\java.

[b]The ***rticonnextmsg*** library only applies if you have the RTI Connext DDS Professional, Evaluation, or Basic package type. It is not provided with the RTI Connext DDS Core package type.

[c]Some library names include a [***version***], which depends on your version of .NET.
For .NET 2.0, omit the [***version***].
For other .NET versions, use the digits, such as 451 or 46. See **Table 12.1 Supported Windows Platforms**.for supported .NET versions.

Table 12.3 Building Instructions for Windows Target Architectures

API	Library Format	RTI Libraries or Jar Files [a] [b] [c]	Required System Libraries	Required Compiler Flags
C++	Static Release	nddscppz.lib nddscz.lib nddscorez.lib rticonnextmsgcppz.lib	netapi32.lib advapi32.lib user32.lib ws2_32.lib	/Gd /EHsc /MT /D “WIN32” /D “RTI_WIN32” /D “NDEBUG”
	Static Debug	nddscppzd.lib nddsczd.lib nddscorezd.lib rticonnextmsgcppzd.lib		/Gd /EHsc /MTd /D “WIN32” /D “RTI_WIN32”
	Dynamic Release	nddscpp.lib nddsc.lib nddscore.lib rticonnextmsgcpp.lib		/Gd /EHsc /MD /D “WIN32” /D “NDDS_DLL_ VARIABLE” /D “RTI_WIN32” /D “NDEBUG”
	Dynamic Debug	nddscppd.lib nddscd.lib nddscored.lib rticonnextmsgcppd.lib		/Gd /EHsc /MDd /D “WIN32” /D “NDDS_DLL_ VARIABLE” /D “RTI_WIN32”
C#	Release	nddsdotnet<*version*>.dll rticonnextmsgdotnet<*version*>.dll	N/A	N/A
	Debug	nddsdotnet<*version*>d.dll rticonnextmsgdotnet<*version*>d.dll		

[a]The RTI C/C++/Java libraries are in <NDDSHOME>\lib\<architecture>. Jar files are in <NDDSHOME>\lib\java.

[b]The ***rticonnextmsg*** library only applies if you have the RTI Connext DDS Professional, Evaluation, or Basic package type. It is not provided with the RTI Connext DDS Core package type.

[c]Some library names include a [***version***], which depends on your version of .NET.
For .NET 2.0, omit the [***version***].
For other .NET versions, use the digits, such as 451 or 46. See **Table 12.1 Supported Windows Platforms**.for supported .NET versions.

Table 12.3 Building Instructions for Windows Target Architectures

API	Library Format	RTI Libraries or Jar Files [a] [b] [c]	Required System Libraries	Required Compiler Flags
C++/CLI	Release	nddscpp.lib nddsc.lib nddscore.lib rticonnextmsgdotnet<*version*>.dll	netapi32.lib advapi32.lib user32.lib ws2_32.lib	/Gd /EHsc /MD /D "WIN32" /D "NDDS_DLL_ VARIABLE" /D "RTI_WIN32" /D "NDEBUG"
	Debug	nddscppd.lib nddscd.lib nddscored.lib rticonnextmsgdotnet<*version*>d.dll		/Gd /EHsc /MDd /D "WIN32" /D "NDDS_DLL_ VARIABLE" /D "RTI_WIN32"
Java	Release	nddsjava.jar rticonnextmsg.jar	N/A	N/A
	Debug	nddsjavad.jar rticonnextmsgd.jar		

Table 12.4 Running Instructions for Windows Architectures

RTI Architecture	Library Format	Environment Variables[d]
All supported Windows architectures for Java	N/A	Path=%NDDSHOME%\lib\<*architecture*>; %Path%
All other supported Windows architectures	Static (Release and Debug)	None required
	Dynamic (Release and Debug)	Path=%NDDSHOME%\lib\<*architecture*>; %Path%

[a] The RTI C/C++/Java libraries are in <NDDSHOME>\lib\<architecture>. Jar files are in <NDDSHOME>\lib\java.

[b] The ***rticonnextmsg*** library only applies if you have the RTI Connext DDS Professional, Evaluation, or Basic package type. It is not provided with the RTI Connext DDS Core package type.

[c] Some library names include a [***version***], which depends on your version of .NET.
For .NET 2.0, omit the [***version***].
For other .NET versions, use the digits, such as 451 or 46. See **Table 12.1 Supported Windows Platforms**.for supported .NET versions.

[d] %Path% represents the value of the Path variable prior to changing it to support Connext DDS. When using nddsjava.jar, the Java virtual machine (JVM) will attempt to load release versions of the native libraries. When using nddsjavad.jar, the JVM will attempt to load debug versions of the native libraries.

Table 12.5 Library-Creation Details for Windows Architectures

RTI Architecture	Library Format	Compiler Flags Used by RTI
All 32-bit Windows architectures for .NET	Dynamic Release	/O2 /GL /D "WIN32" /D "NDEBUG" /D "NDDS_DLL_VARIABLE" /D "_WINDLL" /D "_UNICODE" /D "UNICODE" /FD /EHa /MD /c /Zi /clr /TP
	Dynamic Debug	/Od /D "WIN32" /D "_DEBUG" /D "NDDS_DLL_VARIABLE" /D "_WINDLL" /D "_UNICODE" /D "UNICODE" /FD /EHa /MDd /c /Zi /clr /TP
All 64-bit Windows architectures for .NET	Dynamic Release	/O2 /GL /D "WIN64" /D "NDEBUG" /D "NDDS_DLL_VARIABLE" /D "_WINDLL" /D "_UNICODE" /D "UNICODE" /FD /EHa /MD /c /Zi /clr /TP
	Dynamic Debug	/Od /D "WIN64" /D "_DEBUG" /D "NDDS_DLL_VARIABLE" /D "_WINDLL" /D "_UNICODE" /D "UNICODE" /FD /EHa /MDd /c /Zi /clr /TP
All 32-bit Windows architectures for Java	Dynamic Release	-target 1.4 –source 1.4
	Dynamic Debug	-target 1.4 –source 1.4 -g
All 64-bit Windows architectures for Java	Dynamic Release	-target 1.4 –source 1.6
	Dynamic Debug	-target 1.4 –source 1.6 -g
i86Win32VS2008	Static Release	-DPtrIntType=long -DCSREAL_IS_FLOAT -DCPU=I80586 -DTARGET=\"i86Win32VS2008\" -DWIN32 -D_WINDOWS -D_WIN32_WINNT=0x0501 -DWIN32_LEAN_AND_MEAN /O2 /Zi /MT /EHsc -D_CRT_SECURE_NO_DEPRECATE -DNDEBUG -c
	Dynamic Release	-DPtrIntType=long -DCSREAL_IS_FLOAT -DCPU=I80586 -DTARGET=\"i86Win32VS2008\" -DWIN32 -D_WINDOWS -D_WIN32_WINNT=0x0501 -DWIN32_LEAN_AND_MEAN /O2 /Zi /MD /EHsc -D_CRT_SECURE_NO_DEPRECATE -DNDEBUG -c
	Static Debug	-DPtrIntType=long -DCSREAL_IS_FLOAT -DCPU=I80586 -DTARGET=\"i86Win32VS2008\" -DWIN32 -D_WINDOWS -D_WIN32_WINNT=0x0501 -DWIN32_LEAN_AND_MEAN /Od /ZI /MTd /EHsc /RTC1 -D_CRT_SECURE_NO_DEPRECATE -c
	Dynamic Debug	-DPtrIntType=long -DCSREAL_IS_FLOAT -DCPU=I80586 -DTARGE T=\"i86Win32VS2008\" -DWIN32 -D_WINDOWS -D_WIN32_WINNT=0x0501 -DWIN32_LEAN_AND_MEAN /Od /ZI /MDd /EHsc /RTC1 -D_CRT_SECURE_NO_DEPRECATE -c

Table 12.5 Library-Creation Details for Windows Architectures

RTI Architecture	Library Format	Compiler Flags Used by RTI
i86Win32VS2010	Static Release	-DPtrIntType=long -DCSREAL_IS_FLOAT -DCPU=I80586 -DTARGET=\"i86Win32VS2010\" -DWIN32 -D_WINDOWS -D_WIN32_WINNT=0x0501 -DWIN32_LEAN_AND_MEAN /O2 /Zi /MT /EHsc -D_CRT_SECURE_NO_DEPRECATE -DNDEBUG -c
	Dynamic Release	-DPtrIntType=long -DCSREAL_IS_FLOAT -DCPU=I80586 -DTARGET=\"i86Win32VS2010\" -DWIN32 -D_WINDOWS -D_WIN32_WINNT=0x0501 -DWIN32_LEAN_AND_MEAN /O2 /Zi /MD /EHsc -D_CRT_SECURE_NO_DEPRECATE -DNDEBUG -c
	Static Debug	-DPtrIntType=long -DCSREAL_IS_FLOAT -DCPU=I80586 -DTARGET=\"i86Win32VS2010\" -DWIN32 -D_WINDOWS -D_WIN32_WINNT=0x0501 -DWIN32_LEAN_AND_MEAN /Od /ZI /MTd /EHsc /RTC1 -D_CRT_SECURE_NO_DEPRECATE -c
	Dynamic Debug	-DPtrIntType=long -DCSREAL_IS_FLOAT -DCPU=I80586 -DTARGE T=\"i86Win32VS2010\" -DWIN32 -D_WINDOWS -D_WIN32_WINNT=0x0501 -DWIN32_LEAN_AND_MEAN /Od /ZI /MDd /EHsc /RTC1 -D_CRT_SECURE_NO_DEPRECATE -c
i86Win32VS2012	Static Release	-DPtrIntType=long -DCSREAL_IS_FLOAT -DCPU=I80586 -DTARGET=\"i86Win32VS2012\" -DWIN32 -D_WINDOWS -D_WIN32_WINNT=0x0501 -DWIN32_LEAN_AND_MEAN /O2 /Zi /MT /EHsc -D_CRT_SECURE_NO_DEPRECATE -DNDEBUG -c
	Dynamic Release	-DPtrIntType=long -DCSREAL_IS_FLOAT -DCPU=I80586 -DTARGET=\"i86Win32VS2012\" -DWIN32 -D_WINDOWS -D_WIN32_WINNT=0x0501 -DWIN32_LEAN_AND_MEAN /O2 /Zi /MD /EHsc -D_CRT_SECURE_NO_DEPRECATE -DNDEBUG -c
	Static Debug	-DPtrIntType=long -DCSREAL_IS_FLOAT -DCPU=I80586 -DTARGET=\"i86Win32VS2012\" -DWIN32 -D_WINDOWS -D_WIN32_WINNT=0x0501 -DWIN32_LEAN_AND_MEAN /Od /ZI /MTd /EHsc /RTC1 -D_CRT_SECURE_NO_DEPRECATE -c
	Dynamic Debug	-DPtrIntType=long -DCSREAL_IS_FLOAT -DCPU=I80586 -DTARGE T=\"i86Win32VS2012\" -DWIN32 -D_WINDOWS -D_WIN32_WINNT=0x0501 -DWIN32_LEAN_AND_MEAN /Od /ZI /MDd /EHsc /RTC1 -D_CRT_SECURE_NO_DEPRECATE -c

Table 12.5 Library-Creation Details for Windows Architectures

RTI Architecture	Library Format	Compiler Flags Used by RTI
i86Win32VS2013	Static Release	-DPtrIntType=long -DCSREAL_IS_FLOAT -DCPU=I80586 -DTARGET=\"i86Win32VS2013\" -DWIN32 -D_WINDOWS -D_WIN32_WINNT=0x0501 -DWIN32_LEAN_AND_MEAN /O2 /Zi /MT /EHsc -D_CRT_SECURE_NO_DEPRECATE -DNDEBUG -c
	Dynamic Release	-DPtrIntType=long -DCSREAL_IS_FLOAT -DCPU=I80586 -DTARGET=\"i86Win32VS2013\" -DWIN32 -D_WINDOWS -D_WIN32_WINNT=0x0501 -DWIN32_LEAN_AND_MEAN /O2 /Zi /MD /EHsc -D_CRT_SECURE_NO_DEPRECATE -DNDEBUG -c
	Static Debug	-DPtrIntType=long -DCSREAL_IS_FLOAT -DCPU=I80586 -DTARGET=\"i86Win32VS2013\" -DWIN32 -D_WINDOWS -D_WIN32_WINNT=0x0501 -DWIN32_LEAN_AND_MEAN /Od /ZI /MTd /EHsc /RTC1 -D_CRT_SECURE_NO_DEPRECATE -c
	Dynamic Debug	-DPtrIntType=long -DCSREAL_IS_FLOAT -DCPU=I80586 -DTARGE T=\"i86Win32VS2013\" -DWIN32 -D_WINDOWS -D_WIN32_WINNT=0x0501 -DWIN32_LEAN_AND_MEAN /Od /ZI /MDd /EHsc /RTC1 -D_CRT_SECURE_NO_DEPRECATE -c
i86Win32VS2015	Static Release	-DPtrIntType=long -DCSREAL_IS_FLOAT -DCPU=I80586 -DTARGET=\"i86Win32VS2015\" -DWIN32 -D_WINDOWS -D_WIN32_WINNT=0x0501 -DWIN32_LEAN_AND_MEAN /O2 /Zi /MT /EHsc -D_CRT_SECURE_NO_DEPRECATE -DNDEBUG -c
	Dynamic Release	-DPtrIntType=long -DCSREAL_IS_FLOAT -DCPU=I80586 -DTARGET=\"i86Win32VS2015\" -DWIN32 -D_WINDOWS -D_WIN32_WINNT=0x0501 -DWIN32_LEAN_AND_MEAN /O2 /Zi /MD /EHsc -D_CRT_SECURE_NO_DEPRECATE -DNDEBUG -c
	Static Debug	-DPtrIntType=long -DCSREAL_IS_FLOAT -DCPU=I80586 -DTARGET=\"i86Win32VS2015\" -DWIN32 -D_WINDOWS -D_WIN32_WINNT=0x0501 -DWIN32_LEAN_AND_MEAN /Od /ZI /MTd /EHsc /RTC1 -D_CRT_SECURE_NO_DEPRECATE -c
	Dynamic Debug	-DPtrIntType=long -DCSREAL_IS_FLOAT -DCPU=I80586 -DTARGE T=\"i86Win32VS2015\" -DWIN32 -D_WINDOWS -D_WIN32_WINNT=0x0501 -DWIN32_LEAN_AND_MEAN /Od /ZI /MDd /EHsc /RTC1 -D_CRT_SECURE_NO_DEPRECATE -c

Table 12.5 Library-Creation Details for Windows Architectures

RTI Architecture	Library Format	Compiler Flags Used by RTI
x64Win64VS2008 Note: linker requires /MACHINE:X64 option.	Static Release	/W3 -DPtrIntType=long -DCSREAL_IS_FLOAT -DCPU=AMD64-DTARGET=\"x64Win64VS2008\" -DWIN32 -D_WINDOWS -D_WIN32_WINNT=0x0501 -DWIN32_LEAN_AND_MEAN /O2 /Zi /MT /EHsc -D_CRT_SECURE_NO_DEPRECATE -DNDEBUG -c
	Dynamic Release	/W3 -DPtrIntType=long -DCSREAL_IS_FLOAT -DCPU=AMD64-DTARGET=\"x64Win64VS2008\" -DWIN32 -D_WINDOWS -D_WIN32_WINNT=0x0501 -DWIN32_LEAN_AND_MEAN /O2 /Zi /MD /EHsc -D_CRT_SECURE_NO_DEPRECATE -DNDEBUG -c
	Static Debug	/W3 -DPtrIntType=long -DCSREAL_IS_FLOAT -DCPU=AMD64-DTARGET=\"x64Win64VS2008\" -DWIN32 -D_WINDOWS -D_WIN32_WINNT=0x0501 -DWIN32_LEAN_AND_MEAN /Od /ZI /MTd /EHsc /RTC1 -D_CRT_SECURE_NO_DEPRECATE -c
	Dynamic Debug	/W3 -DPtrIntType=long -DCSREAL_IS_FLOAT -DCPU=AMD64-DTARGET=\"x64Win64VS2008\" -DWIN32 -D_WINDOWS -D_WIN32_WINNT=0x0501 -DWIN32_LEAN_AND_MEAN /Od /ZI /MDd /EHsc /RTC1 -D_CRT_SECURE_NO_DEPRECATE -c
x64Win64VS2010 Note: linker requires /MACHINE:X64 option.	Static Release	/W3 -DPtrIntType=long -DCSREAL_IS_FLOAT -DCPU=AMD64-DTARGET=\"x64Win64VS2010\" -DWIN32 -D_WINDOWS -D_WIN32_WINNT=0x0501 -DWIN32_LEAN_AND_MEAN /O2 /Zi /MT /EHsc -D_CRT_SECURE_NO_DEPRECATE -DNDEBUG -c
	Dynamic Release	/W3 -DPtrIntType=long -DCSREAL_IS_FLOAT -DCPU=AMD64-DTARGET=\"x64Win64VS2010\" -DWIN32 -D_WINDOWS -D_WIN32_WINNT=0x0501 -DWIN32_LEAN_AND_MEAN /O2 /Zi /MD /EHsc -D_CRT_SECURE_NO_DEPRECATE -DNDEBUG -c
	Static Debug	/W3 -DPtrIntType=long -DCSREAL_IS_FLOAT -DCPU=AMD64-DTARGET=\"x64Win64VS2010\" -DWIN32 -D_WINDOWS -D_WIN32_WINNT=0x0501 -DWIN32_LEAN_AND_MEAN /Od /ZI /MTd /EHsc /RTC1 -D_CRT_SECURE_NO_DEPRECATE -c
	Dynamic Debug	/W3 -DPtrIntType=long -DCSREAL_IS_FLOAT -DCPU=AMD64-DTARGET=\"x64Win64VS2010\" -DWIN32 -D_WINDOWS -D_WIN32_WINNT=0x0501 -DWIN32_LEAN_AND_MEAN /Od /ZI /MDd /EHsc /RTC1 -D_CRT_SECURE_NO_DEPRECATE -c

Table 12.5 Library-Creation Details for Windows Architectures

RTI Architecture	Library Format	Compiler Flags Used by RTI
x64Win64VS2012 Note: linker requires /MACHINE:X64 option.	Static Release	/W3 -DPtrIntType=long -DCSREAL_IS_FLOAT -DCPU=AMD64-DTARGET=\"x64Win64VS2012\" -DWIN32 -D_WINDOWS -D_WIN32_WINNT=0x0501 -DWIN32_LEAN_AND_MEAN /O2 /Zi /MT /EHsc -D_CRT_SECURE_NO_DEPRECATE -DNDEBUG -c
	Dynamic Release	/W3 -DPtrIntType=long -DCSREAL_IS_FLOAT -DCPU=AMD64-DTARGET=\"x64Win64VS2012\" -DWIN32 -D_WINDOWS -D_WIN32_WINNT=0x0501 -DWIN32_LEAN_AND_MEAN /O2 /Zi /MD /EHsc -D_CRT_SECURE_NO_DEPRECATE -DNDEBUG -c
	Static Debug	/W3 -DPtrIntType=long -DCSREAL_IS_FLOAT -DCPU=AMD64-DTARGET=\"x64Win64VS2012\" -DWIN32 -D_WINDOWS -D_WIN32_WINNT=0x0501 -DWIN32_LEAN_AND_MEAN /Od /ZI /MTd /EHsc /RTC1 -D_CRT_SECURE_NO_DEPRECATE -c
	Dynamic Debug	/W3 -DPtrIntType=long -DCSREAL_IS_FLOAT -DCPU=AMD64-DTARGET=\"x64Win64VS2012\" -DWIN32 -D_WINDOWS -D_WIN32_WINNT=0x0501 -DWIN32_LEAN_AND_MEAN /Od /ZI /MDd /EHsc /RTC1 -D_CRT_SECURE_NO_DEPRECATE -c
x64Win64VS2013 Note: linker requires /MACHINE:X64 option.	Static Release	/W3 -DPtrIntType=long -DCSREAL_IS_FLOAT -DCPU=AMD64-DTARGET=\"x64Win64VS2013\" -DWIN32 -D_WINDOWS -D_WIN32_WINNT=0x0501 -DWIN32_LEAN_AND_MEAN /O2 /Zi /MT /EHsc -D_CRT_SECURE_NO_DEPRECATE -DNDEBUG -c
	Dynamic Release	/W3 -DPtrIntType=long -DCSREAL_IS_FLOAT -DCPU=AMD64-DTARGET=\"x64Win64VS2013\" -DWIN32 -D_WINDOWS -D_WIN32_WINNT=0x0501 -DWIN32_LEAN_AND_MEAN /O2 /Zi /MD /EHsc -D_CRT_SECURE_NO_DEPRECATE -DNDEBUG -c
	Static Debug	/W3 -DPtrIntType=long -DCSREAL_IS_FLOAT -DCPU=AMD64-DTARGET=\"x64Win64VS2013\" -DWIN32 -D_WINDOWS -D_WIN32_WINNT=0x0501 -DWIN32_LEAN_AND_MEAN /Od /ZI /MTd /EHsc /RTC1 -D_CRT_SECURE_NO_DEPRECATE -c
	Dynamic Debug	/W3 -DPtrIntType=long -DCSREAL_IS_FLOAT -DCPU=AMD64-DTARGET=\"x64Win64VS2013\" -DWIN32 -D_WINDOWS -D_WIN32_WINNT=0x0501 -DWIN32_LEAN_AND_MEAN /Od /ZI /MDd /EHsc /RTC1 -D_CRT_SECURE_NO_DEPRECATE -c

Table 12.5 Library-Creation Details for Windows Architectures

RTI Architecture	Library Format	Compiler Flags Used by RTI
x64Win64VS2015 Note: linker requires /MACHINE:X64 option.	Static Release	/W3 -DPtrIntType=long -DCSREAL_IS_FLOAT -DCPU=AMD64-DTARGET=\"x64Win64VS2015\" -DWIN32 -D_WINDOWS -D_WIN32_WINNT=0x0501 -DWIN32_LEAN_AND_MEAN /O2 /Zi /MT /EHsc -D_CRT_SECURE_NO_DEPRECATE -DNDEBUG -c
	Dynamic Release	/W3 -DPtrIntType=long -DCSREAL_IS_FLOAT -DCPU=AMD64-DTARGET=\"x64Win64VS2015\" -DWIN32 -D_WINDOWS -D_WIN32_WINNT=0x0501 -DWIN32_LEAN_AND_MEAN /O2 /Zi /MD /EHsc -D_CRT_SECURE_NO_DEPRECATE -DNDEBUG -c
	Static Debug	/W3 -DPtrIntType=long -DCSREAL_IS_FLOAT -DCPU=AMD64-DTARGET=\"x64Win64VS2015\" -DWIN32 -D_WINDOWS -D_WIN32_WINNT=0x0501 -DWIN32_LEAN_AND_MEAN /Od /ZI /MTd /EHsc /RTC1 -D_CRT_SECURE_NO_DEPRECATE -c
	Dynamic Debug	/W3 -DPtrIntType=long -DCSREAL_IS_FLOAT -DCPU=AMD64-DTARGET=\"x64Win64VS2015\" -DWIN32 -D_WINDOWS -D_WIN32_WINNT=0x0501 -DWIN32_LEAN_AND_MEAN /Od /ZI /MDd /EHsc /RTC1 -D_CRT_SECURE_NO_DEPRECATE -c

12.1 Requirements when Using Microsoft Visual Studio

When Using Visual Studio 2008 — Service Pack 1 Requirement

You must have Visual Studio 2008 Service Pack 1 or the Microsoft Visual C++ 2008 SP1 Redistributable Package installed on the machine where you are *running* an application linked with dynamic libraries.

This includes dynamically linked C/C++ and all .NET and Java applications. The Microsoft Visual C++ 2008 SP1 Redistributable Package can be downloaded from the following Microsoft websites:

For x86 architectures:

http://www.microsoft.com/downloads/details.aspx?familyid=A5C84275-3B97-4AB7-A40D-3802B2AF5FC2&displaylang=en

For x64 architectures:

http://www.microsoft.com/downloads/details.aspx?FamilyID=ba9257ca-337f-4b40-8c14-157cf-dffee4e&displaylang=en

When Using Visual Studio 2010 — Service Pack 1 Requirement

You must have Visual Studio 2010 Service Pack 1 or the Microsoft Visual C++ 2010 SP1 Redistributable Package installed on the machine where you are *running* an application linked with dynamic libraries.

This includes dynamically linked C/C++ and all .NET and Java applications. To run an application built with debug libraries of the above RTI architecture packages, you must have Visual Studio 2010 Service Pack 1 installed.

The Microsoft Visual C++ 2010 Service Pack 1 Redistributable Package can be obtained from the following Microsoft websites:

For x86 architectures:https://www.microsoft.com/en-us/download/details.aspx?id=8328

For x64 architectures: https://www.microsoft.com/en-us/download/details.aspx?id=13523

When Using Visual Studio 2012 — Update 4 Redistributable Package Requirement

You must have the Visual C++ Redistributable for Visual Studio 2012 Update 4 installed on the machine where you are *running* an application linked with dynamic libraries. This includes dynamically linked C/C++ and all .NET and Java applications.

You can download Visual C++ Redistributable for Visual Studio 2012 Update 4 from this Microsoft website: http://www.microsoft.com/en-ca/download/details.aspx?id=30679

When Using Visual Studio 2013 — Update 4 Redistributable Package Requirement

You must have Visual C++ Redistributable for Visual Studio 2013 Update 4 installed on the machine where you are *running* an application linked with dynamic libraries. This includes C/C++ dynamically linked and all .NET and Java applications.

You can download Visual C++ Redistributable for Visual Studio 2013 Update 4 from this Microsoft website: https://www.microsoft.com/en-us/download/details.aspx?id=40784

When Using Visual Studio 2015 — Update 1 Redistributable Package Requirement

You must have Visual C++ Redistributable for Visual Studio 2015 Update 1 installed on the machine where you are running an application linked with dynamic libraries. This includes C/C++ dynamically linked and all .NET and Java applications.

You can download Visual C++ Redistributable for Visual Studio 2015 Update 1 from this Microsoft website: https://www.microsoft.com/en-us/download/details.aspx?id=49984

12.2 Windows Registry Setting for Better Performance

On all Windows systems *prior to* Windows Vista, the following registry setting change will improve performance when sending UDP datagrams of size larger than 1024 bytes:

Under **HKEY_LOCAL_MACHINE, SYSTEM, CurrentControlSet, Services, AFD, Parameters**, add the following:

```
DWORD: Name=FastSendDatagramThreshold, Value = 65536
```

This will improve the Connext DDS performance for data sizes larger than 1024 bytes (RTPS overhead included). It allows the datagrams to bypass the I/O subsystem by using a blocking send call instead of a buffer copy in the Windows Network stack.

12.3 Use Dynamic MFC Library, Not Static

To avoid communication problems in your Connext DDS application, use the dynamic MFC library, not the static version.

If you use the static version, your Connext DDS application may stop receiving DDS samples once the Windows sockets are initialized.

12.4 .NET API Requires Thread Affinity

To maintain proper concurrency control, .NET threads that call a Connext DDS API must correspond one-to-one with operating system threads. In most applications, this will always be the case. However, it may not be the case if the threads you are using are managed in a more advanced way—for example, Microsoft SQL Server does this, or you may do so in your own application.

If you intend to call Connext DDS APIs from explicitly managed threads, you must first call **Thread.BeginThreadAffinity()** in each such thread to ensure that it remains attached to a single operating system thread. See http://msdn.microsoft.com/en-us/library/system.threading.thread.beginthreadaffinity.aspx.

Note: When done making RTI calls from a given thread, call **Thread.EndThreadAffinity()**.

In any case, be sure to consult the RTI API documentation for more information about the thread safety contracts of the operations you use.

12.5 ODBC Database Compatibility

To use the Durable Writer History and Durable Reader State features, you must install a relational database such as MySQL.

In principle, you can use any database that provides an ODBC driver, since ODBC is a standard. However, not all ODBC databases support the same feature set. Therefore, there is no guarantee that the persistent durability features will work with an arbitrary ODBC driver.

We have tested the following driver:

- MySQL ODBC 5.1.44

Note: Starting with 4.5e, support for the TimesTen database has been removed.

To use MySQL, you also need the MySQL ODBC 5.1.6 (or higher) driver.

The Durable Writer History and Durable Reader State features have been tested with the following architectures:

- i86Win32VS2008
- i86Win32VS2010
- x64Win64VS2008
- x64Win64VS2010

For more information on database setup, please see the *RTI Connext DDS Core Libraries Getting Started Guide Addendum for Database Setup*.

12.6 PPP Link Support for Windows XP Systems

To use a Windows XP point-to-point protocol (PPP) link (such as a serial cable), the UDP transport properties for the Connext DDS applications running on the PPP server machine *must* be configured with multicast disabled for the PPP server interface(s).

To disable multicast for an interface, change the UDPv4 transport properties as follows:

```
// Disable multicast for PPP interface because it causes problems:
char *bad_interfaces[] = { "192.168.250.100"}; // interface addr
const int num_bad_interfaces =
       sizeof(bad_interfaces)/sizeof(bad_interfaces[0]);
UDPv4Properties.parent.deny_multicast_interfaces_list =
    bad_interfaces;
UDPv4Properties.parent.deny_multicast_interfaces_list_length =
    num_bad_interfaces;
```

Failure to do so will result in Connext DDS being unable to send any data at all over the PPP link.

Notes:

- Setting up multicast-related socket options for the PPP interface can prevent future *unicast* sends using that socket from working.
- Connext DDS sets up certain sockets for multicast even if it has no multicast peers, in case some show up later. You avoid this by configuring the multicast deny list as described above.

12.7 Support for Modern C++ API

Connext DDS provides two different C++ APIs, which we refer to as the "Traditional C++" and "Modern C++" APIs. The Modern C++ API is available for all Windows platforms.

For more information on Modern C++, see "Traditional vs. Modern C++" in the User's Manual.

12.8 Multicast Support

Multicast is supported on all platforms and is configured out of the box. That is, the default value for the initial peers list (**NDDS_DISCOVERY_PEERS**) includes a multicast address. See the online documentation for more information.

12.9 Supported Transports

Shared memory: Shared memory is supported and enabled by default. The Windows operating system manages the shared memory resources automatically. Cleanup is not required.

UDPv4: Supported and enabled by default.

UDPv6: Supported but disabled on architectures that use Visual Studio. The peers list (**NDDS_DISCOVERY_PEERS**) must be modified to support UDPv6. No Traffic Class support.

TCP/IPv4: Supported on architectures that use Visual Studio. (This is *not* a built-in transport.)

12.10 Monotonic Clock Support

The monotonic clock (described in "Clock Selection" in the User's Manual) is supported on all Windows platforms.

12.11 Thread Configuration

Thread Settings for Windows Platforms (Section Table 12.6 on the next page) lists the thread settings for Windows platforms.

Thread-Priority Definitions for Windows Platforms (Section Table 12.7 on page 158) and Thread Kinds for Windows Platforms (Section Table 12.8 on page 158) list the thread-priority definitions and thread kinds, respectively.

Table 12.6 Thread Settings for Windows Platforms

Applicable Thread	DDS_ ThreadSettings_t	Platform-Specific Setting
Asynchronous Publisher, Asynchronous flushing thread,	mask	OS default thread type
	priority	0
	stack_size	OS default thread stack size
	cpu_list	CPU core affinity not supported
	cpu_rotation	CPU core affinity not supported
Database thread	mask	DDS_THREAD_SETTINGS_STDIO
	priority	-3
	stack_size	OS default thread stack size
	cpu_list	CPU core affinity not supported
	cpu_rotation	CPU core affinity not supported
Event thread	mask	DDS_THREAD_SETTINGS_STDIO | DDS_THREAD_SETTINGS_ FLOATING_POINT
	priority	-2
	stack_size	OS default thread stack size
	cpu_list	CPU core affinity not supported
	cpu_rotation	CPU core affinity not supported
ReceiverPool threads	mask	DDS_THREAD_SETTINGS_STDIO | DDS_THREAD_SETTINGS_ FLOATING_POINT
	priority	2
	stack_size	OS default thread stack size
	cpu_list	CPU core affinity not supported
	cpu_rotation	CPU core affinity not supported

Table 12.7 Thread-Priority Definitions for Windows Platforms

Thread-Priority Definition	Operating-System Priority
THREAD_PRIORITY_DEFAULT	0
THREAD_PRIORITY_HIGH	3
THREAD_PRIORITY_ABOVE_NORMAL	2
THREAD_PRIORITY_NORMAL	0
THREAD_PRIORITY_BELOW_NORMAL	-2
THREAD_PRIORITY_LOW	-3

Table 12.8 Thread Kinds for Windows Platforms

Thread Kinds	Operating-System Configuration[a]
DDS_THREAD_SETTINGS_FLOATING_POINT	N/A
DDS_THREAD_SETTINGS_STDIO	
DDS_THREAD_SETTINGS_REALTIME_PRIORITY	
DDS_THREAD_SETTINGS_PRIORITY_ENFORCE	

12.11.1 Support for Controlling CPU Core Affinity for RTI Threads

Support for controlling CPU core affinity (described in "Controlling CPU Core Affinity" in the User's Manual) is not available for Windows platforms.

12.12 Durable Writer History and Durable Reader State Features

The Durable Writer History and Durable Reader State features are only supported on platforms that use 32-bit/64-bit Visual Studio 2008 and Visual Studio 2010.

[a]See Windows manuals for additional information.

12.13 Libraries Required for Using Distributed Logger Support

RTI Distributed Logger is supported on all Windows platforms. Table 12.9 Additional Libraries for using RTI Distributed Logger lists the additional libraries you will need in order to use *Distributed Logger*.

Table 12.9 Additional Libraries for using RTI Distributed Logger

Language	Static		Dynamic	
	Release	Debug	Release	Debug
C	rtidlcz.lib	rtidlczd.lib	rtidlc.dll	rtidlcd.dll
C++ (Traditional API)	rtidlcz.lib rtidlcppz.lib	rtidlczd.lib rtidlcppzd.lib	rtidlc.dll rtidlcpp.dll	rtidlcd.dll rtidlcppd.dll
Java	N/A	N/A	distlog.jar distlogdatamodel.jar	distlogd.jar distlogdatamodeld.jar

12.14 Libraries Required for Using Monitoring

Make sure you are consistent in your use of static, dynamic, debug and release versions of the libraries. For example, if your Connext DDS application is linked with the static release version of the Connext DDS libraries, you will need to also use the static release version of the monitoring library. Do not link both static and dynamic libraries. Similarly, do not mix release and debug libraries.

If you are statically linking your application with DDS libraries and you want to add monitoring to your application, you will also need to statically link the monitoring library. The library cannot be loaded dynamically strictly through the QoS profile because it also depends on DDS to publish its data. Therefore, it depends on DDS; the DDS functionality would cause duplicate symbols to be found resulting, in the termination of the process.

Table 12.10 Additional Libraries for Using Monitoring

Static Release	Static Debug	Dynamic Release	Dynamic Debug
rtimonitoringz.lib Psapi.lib	rtimonitoringzd.lib Psapi.lib	rtimonitoring.lib rtimonitoring.dll	rtimonitoringd.lib rtimonitoringd.dll

12.15 Libraries Required for Using RTI Secure WAN Transport APIs

To use the Secure WAN Transport APIs, add the libraries from Table 12.11 Additional Libraries for Using RTI Secure WAN Transport APIs on Windows Systems to your project files.

Table 12.11 Additional Libraries for Using RTI Secure WAN Transport APIs on Windows Systems

Library Format	RTI Secure WAN Transport Libraries[a]	OpenSSL Libraries[b]
Dynamic Release	nddstransportwan.lib nddstransporttls.lib	ssleay32.lib libeay32.lib
Dynamic Debug	nddstransporttlsd.lib nddstransportwand.lib	
Static Release	nddstransportwanz.lib nddstransporttlsz.lib	
Static Debug	nddstransportwanzd.lib nddstransporttlszd.lib	

12.16 Libraries Required for Using RTI TCP Transport APIs

To use the TCP Transport APIs, link against the additional libraries from Table 12.12 Additional Libraries for Using RTI TCP Transport APIs on Windows Systems or Table 12.13 Additional Libraries for using RTI TCP Transport APIs on Windows Systems with TLS Enabled. (Select the files appropriate for your chosen library format.)

Table 12.12 Additional Libraries for Using RTI TCP Transport APIs on Windows Systems

Library Format	RTI TCP Transport Libraries[c]
Dynamic Release	nddstransporttcp.dll
Dynamic Debug	nddstransporttcpd.dll
Static Release	nddstransporttcpz.lib
Static Debug	nddstransporttcpzd.lib

[a]These libraries are in <<NDDSHOME>\lib\<architecture>

[b]These libraries are in <openssl install dir>\<architecture>/lib, where <openssl install dir> is where OpenSSL is installed

[c]The libraries are in <NDDSHOME>\lib\<architecture>

Table 12.13 Additional Libraries for using RTI TCP Transport APIs on Windows Systems with TLS Enabled

Library Format	RTI TLS Libraries[a]
Dynamic Release	nddstls.dll
Dynamic Debug	nddstlsd.dll
Static Release	nddstlsz.dll
Static Debug	nddstlszd.dll
OpenSSL Libraries	ssleay32.lib libeay32.lib

[a]The libraries are in <NDDSHOME>\lib\<architecture>