RTI Connext DDS Core Libraries

Platform Notes

Version 6.0.1

Printed in U.S.A. First printing.
November 2019.

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/

Contents

Chapter 1 Supported Platforms

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

For each platform, this document provides information on:

- Supported operating systems and compilers
- Required *Connext DDS* and system libraries
- Required compiler and linker flags, and environment variables for running the application (if any)
- Details on how the *Connext DDS* libraries were built
- Modern C++ API support
- Multicast support
- Supported transports
- Monotonic clock support
- Thread configuration
- Durable Writer History and Durable Reader State features support
- Support for 'Find Package' CMake script
- Required libraries for using additional features (such as Distributed Logger, Monitoring, Secure WAN, TLS, Zero Copy transfer over shared memory)

Table 1.1 Supported Platforms

Operating System	Version
AIX®	AIX 7.1
Android™ (target only)	Android 5.0, 5.1, 9.0 (5.0 and 5.1 only available on request.)

Table 1.1 Supported Platforms

Operating System	Version
INTEGRITY® (target only)	INTEGRITY 10.0.2, 11.0.4, 11.4.4 (10.0.2 and 11.0.4 only available on request.)
iOS® (target only)	iOS 8.2 (Only available on request.)
Linux® (Arm® CPU)	NI™ Linux 3 Raspbian Wheezy 7.0 Ubuntu® 16.04 LTS
Linux (Intel® CPU)	CentOS™ 6.0, 6.2 - 6.4, 7.0 Red Hat® Enterprise Linux 6.0 - 6.5, 6.7, 6.8, 7.0, 7.3, 7.5, 8.0 Ubuntu 12.04 LTS, 14.04 LTS, 16.04 LTS, 18.04 LTS SUSE® Linux Enterprise Server 11 SP2, 11 SP3, 12 SP2 Wind River® Linux 7
LynxOS® (target only)	LynxOS 5.0
macOS®	macOS 10.12 - 10.14 (10.12 only available on request.)
QNX® (target only)	QNX Neutrino® 6.4.1, 6.5, 7.0 (6.4.1 and 6.5 only available on request.)
Solaris™	Solaris 2.10 (Only available on request.)
VxWorks® (target only)	VxWorks 6.9, 6.9.3.2, 6.9.4.2, 6.9.4.6, 7.0 (6.9 only available on request.) VxWorks 653 2.3 (Only available on request.)
Windows®	Windows 8, 8.1, 10[a] Windows Server 2012 R2 Windows Server 2016

The following table lists additional target libraries available for *Connext DDS*, for which RTI offers custom support. If you are interested in using one of these platforms, please contact your local RTI sales representative or email **sales@rti.com**.

Table 1.2 Custom Supported Platforms

Operating System	Version
INTEGRITY	INTEGRITY 5.0.11 on MPC 8349 CPU

[a]Per Microsoft, this should be compatible with Windows 10 IoT Enterprise with Windows native app.

Table 1.2 Custom Supported Platforms

Operating System	Version
INtime® for Windows	INtime 6.3 with Visual Studio 2017 on x86 CPU[a]
Linux	Debian® 7 on Arm v7 CPU
	Freescale™ 1.4 on QorIQ or P4040/P4080/P4081 CPU
	Red Hat Enterprise Linux 5.2 on x86 CPU
	RedHawk™ Linux 6.0 on x64 CPU RedHawk Linux 6.5 on x86 and x64 CPUs
	Wind River Linux 7 on Arm v7 CPU Wind River Linux 8 on PPC e6500 and Arm v7 CPUs
	Xilinx® 14.2 on Arm v7 CPU
	Yocto Project® 2.2 on 64-bit Arm v8 CPU Yocto Project 2.5 on Arm v7 CPU
QNX	QNX 6.5 on PPC e500v2 CPU
	QNX 6.6 on Arm v7 and x86 CPUs
	QNX 7.0 on Arm v7 CPU
VxWorks	VxWorks 6.9.3 on Arm v7 and PPC CPUs VxWorks 6.9.3.2 on MIPS CPU VxWorks 6.9.4.11 on Arm v7 CPU VxWorks 653 2.5.0.2 on PPC e500v2 CPU VxWorks 7.0 SR0540 on x86 CPU

[a]Tested on 64-bit Windows 10 operating system.

1.1 Paths Mentioned in Documentation

The documentation refers to:

- **<NDDSHOME>**

  This refers to the installation directory for *RTI® Connext® DDS*. The default installation paths are:
  - macOS® systems:
    /Applications/rti_connext_dds-6.0.1
  - UNIX-based systems, non-*root* user:
    /home/<*your user name*>/rti_connext_dds-6.0.1
  - UNIX-based systems, *root* user:
    /opt/rti_connext_dds-6.0.1
  - Windows® systems, user without Administrator privileges:
    <*your home directory*>\rti_connext_dds-6.0.1
  - Windows systems, user with Administrator privileges:
    C:\Program Files\rti_connext_dds-6.0.1

  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-6.0.1\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:
  - macOS systems: **/Users/<*your user name*>/rti_workspace/6.0.1/examples**
  - UNIX-based systems: **/home/<*your user name*>/rti_workspace/6.0.1/examples**

- Windows systems: **<*your Windows documents folder*>\rti_workspace\6.0.1\examples**

  Where 'your Windows documents folder' depends on your version of Windows. For example, on Windows 10, the folder is **C:\Users\<*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 *RTI Connext DDS Installation 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 7.1	POWER7 (32-bit mode)	IBM xlC_r for AIX v12.1	p7AIX7.1xlc12.1
		IBM Java 1.8	
	POWER7 (64-bit mode)	IBM xlC_r for AIX v12.1	64p7AIX7.1xlc12.1
		IBM Java 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:

- 2.6 Libraries Required for Using Monitoring on page 14
- 2.7 Libraries Required for Zero Copy Transfer Over Shared Memory 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[abc]	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 or librticonnextmsgcpp2z.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 or librticonnextmsgcpp2zd.a		
	Dynamic Release	libnddscpp.so or libnddscpp2.so libnddsc.so libnddscore.so librticonnextmsgcpp.so or librticonnextmsgcpp2.so	-ldl -lnsl -lm -pthread -brtl	
	Dynamic Debug	libnddscppd.so or libnddscpp2d.so libnddscd.so libnddscored.so librticonnextmsgcppd.so or librticonnextmsgcpp2d.so		

[a]Choose *cpp*.* for the Traditional C++ API or *cpp2*.* for the Modern C++ API.

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

[c]The ***rticonnextmsg*** library only applies if you have the RTI *Connext DDS* Professional, Secure, Basic, or Evaluation 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[abc]	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 *cpp*.* for the Traditional C++ API or *cpp2*.* for the Modern C++ API.

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

[c]The ***rticonnextmsg*** library only applies if you have the RTI *Connext DDS* Professional, Secure, Basic, or Evaluation 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

Table 2.4 Library-Creation Details for AIX Architectures

RTI Architecture	Library Format (Static & Dynamic)	Compiler Flags Used by RTI [c]	Libraries Used by RTI [d]
p7AIX7.1xlc12.1	Release	-q32 -qwarn64 -qlongdouble -qalias=noansi -qpic=large -qthreaded -D_POSIX_C SOURCE-E=199506L -D EXTENSIONS -O -qflag=i:i -DPtrIntType=long -DCSREAL_IS_FLOAT -DCPU-U=Power7+ -DNDEBUG	-lC128
	Debug	-q32 -qwarn64 qlongdouble -qalias=noansi -qpic=large -qthreaded -D_POSIX_C_SOURCE-E=199506L -D EXTENSIONS -O -qflag=i:i -DPtrIntType=long -DCSREAL_IS_FLOAT -DCPU-U=Power7+ -g	-lC128

[a] See 2.3.1 Notes for Using Shared Memory 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).

[c] *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**.

[d] 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-E=199506L -D EXTENSIONS -O -qflag=i:i -DPtrIntType=long -DCSREAL_IS_FLOAT -DCPU-U=Power7+ -DNDEBUG	-lC128
	Debug	-q64 -qwarn64 qlongdouble -qalias=noansi -qpic=large -qthreaded -D_POSIX_C_SOURCE-E=199506L -D EXTENSIONS -O -qflag=i:i -DPtrIntType=long -DCSREAL_IS_FLOAT -DCPU-U=Power7+ -g	-lC128
All supported AIX architectures for Java	Release	-target 1.5 -source 1.5	
	Debug	-target 1.5 -source 1.5 -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 and has been tested with C++03.

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 Unsupported Features

These features are not supported on AIX platforms:

- Controlling CPU Core Affinity
- Distributed Logger
- Durable Writer History and Durable Reader State
- 'Find Package' CMake script
- Monotonic clock
- Internal setting of thread names at the operating-system level

2.5 Thread Configuration

See Table 2.5 Thread Settings for AIX Platforms and Table 2.6 Thread-Priority Definitions 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
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	

2.5.1 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**).

2.6 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.

Notes:

- Memory and CPU usage is not available in monitoring data.
- 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

2.7 Libraries Required for Zero Copy Transfer Over Shared Memory

To use the Zero Copy transfer over shared memory feature, link against the additional library in Table 2.8 Additional Libraries for Zero Copy Transfer Over Shared Memory.

Table 2.8 Additional Libraries for Zero Copy Transfer Over Shared Memory

Static Release	Static Debug	Dynamic Release	Dynamic Debug
libnddsmetpz.a	libnddsmetpzd.a	libnddsmetp.so	libnddsmetpd.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[a]	RTI Architecture Abbreviation
Android 5.0, 5.1	Arm v7A	gcc 4.9 (NDK r10e)[b]	armv7aAndroid5.0gcc4.9ndkr10e
		Java Platform, Standard Edition JDK 1.8	
Android 9.0	Arm v7A	Clang 8.0 (NDK r19b)	armv7Android9.0clang8.0ndkr19b
		Java Platform, Standard Edition JDK 1.8	
	Arm v8A 64 bit	Clang 8.0 (NDK r19b)	arm64Android9.0clang8.0ndkr19b
		Java Platform, Standard Edition JDK 1.8	

See Table 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:

- 3.8 Libraries Required for Using Monitoring on page 24
- 3.9 Libraries Required for Using RTI Secure WAN Transport APIs on page 25
- 3.10 Libraries Required for Using RTI TCP Transport and TLS Support APIs on page 25

---

[a]For Java: Dalvik VM is JDK 1.5 with some features from 1.6 (See Android documentation for details.)

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

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 the 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 librticonnextmsgcppz.a For Android 3 and 5: libgnustl_shared.a For android 9: libc++_shared.a	For Android 5.0 and 5.1 platforms: -L$(SYSROOT)/usr/lib -llog -lm -lc -lgnustl_shared For Android 9 in armv7-a platforms: -sysroot=$(ANDROID_NDK_PATH)/ platforms/android28/arch-arm -LL$(ANDROID_NDK_ PATH)/toolchains/llvm/prebuilt/linux-x86_64/ sysroot/usr/lib/arm-linux-androideabi/28 -llog -lc -lm -L$(ANDROID_NDK_PATH)/sources/cxx-stl/ llvm-libc++/libs/armeabi-v7a -lc++_shared [d] For Android 9.0 in armv8-a platforms: -sysroot=$(ANDROID_NDK_PATH)/ platforms/android28/arch-arm -LL$(ANDROID_NDK_ PATH)/toolchains/llvm/prebuilt/linux-x86_64/ sysroot/usr/lib/arm-linux-androideabi/28 -llog -lc -lm -L$(ANDROID_NDK_PATH)/sources/cxx-stl/ llvm-libc++/libs/arm64-v8a- -lc++_shared [e]	Android 5.0 and 5.1 platforms: -march=armv7-a -mfloat-abi=softfp -mfpu=vfpv3-d16 -mlong-calls -DRTI_UNIX -DRTI_ANDROID Android 9 in armv7-a platforms: -march=armv7-a -DRTI_UNIX -DRTI_ANDROID -DRTI_ANDROID9 Android 9 in armv8-a platforms: -march=arm64v8-a -DRTI_UNIX -DRTI_ANDROID -DRTI_ANDROID9 -DRTI_64BIT
	Static Debug	libnddscppzd.a or libnddscpp2zd.a libnddsczd.a libnddscorezd.a librticonnextmsgcppzd.a For Android 3 and 5: libgnustl_shared.a For android 9: libc++_shared.a		
	Dynamic Release	libnddscpp.so or libnddscpp2.so libnddsc.so libnddscore.so librticonnextmsgcpp.so For Android 3 and 5: libgnustl_shared.so For android 9: libc++_shared.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.

[d]The library **libc++_shared.so** is always required when using C++ RTI Connext libraries and must be included in your application APK.

[e]The library **libc++_shared.so** is always required when using C++ RTI Connext libraries and must be included in your application APK.

Table 3.2 Building Instructions for Android Architectures

API	Library Format	Required RTI Libraries and JAR Files[abc]	Required System Libraries	Required Compiler Flags
	Dynamic Debug	libnddscppd.so or libnddscpp2d.so libnddscd.so libnddscored.so librticonnextmsgcppd.so For Android 3 and 5: libgnustl_shared.so For android 9: libc++_shared.so		
C	Static Release	libnddscz.a libnddscorez.a librticonnextmsgcz.a	For Android 5.0 and 5.1 platforms: -L$(SYSROOT)/usr/lib -llog -lm -lc For Android 9 in armv7-a platforms: -sysroot=$(ANDROID_NDK_PATH)/platforms/android28/arch-arm -LL$(ANDROID_NDK_PATH)/toolchains/llvm/prebuilt/linux-x86_64/sysroot/usr/lib/arm-linuxandroideabi/28 -llog -lc -lm -L$(ANDROID_NDK_PATH)/sources/cxx-stl/llvm-libc++/libs/armeabi-v7a For Android 9 in armv8-a platforms: -sysroot=$(ANDROID_NDK_PATH)/platforms/android28/arch-arm64 -LL$(ANDROID_NDK_PATH)/toolchains/llvm/prebuilt/linux-x86_64/sysroot/usr/lib/arm-linuxandroideabi/28 -llog -lc -lm -L$(ANDROID_NDK_PATH)/sources/cxx-stl/llvm-libc++/libs/arm64-v8a	Android 5.0 and 5.1 platforms: -march=armv7-a -mfloat-abi=softfp -mfpu=vfpv3-d16 -mlong-calls -DRTI_UNIX -DRTI_ANDROID Android 9 in armv7-a platforms: -march=armv7-a -DRTI_UNIX -DRTI_ANDROID -DRTI_ANDROID9 Android 9 in armv8-a platforms: -march=arm64v8-a -DRTI_UNIX -DRTI_ANDROID -DRTI_ANDROID9 -DRTI_64BIT
	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>.

[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
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		

Table 3.3 Running Instructions for Android Architectures

RTI Architecture	Library Format	Required Environment Variables
All supported Android architectures when not using Java	App (*.apk)	None
	Static	None
	Dynamic	LD_LIBRARY_PATH=$LD_LIBRARY_PATH: <path-to-ndds-libs>
All supported Android architectures when using Java	App (*.apk)	None
	Dex	LD_LIBRARY_PATH=$LD_LIBRARY_PATH: <path-to-ndds-libs> CLASSPATH=<path-to-dex>/classes.dex

[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 3.4 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
armv7Android9.0clang8.0ndkr19b	Release	-fpic -DLINUX -O -Wall -Wno-unknown-pragmas -Wno-address -Wno-unused-but-setvariable -DPtrIntType=long -DCSREAL_IS_FLOAT -DCPU=ARMV7 -DTARGET=\"armv7Android9.0clang8.0ndkr19b\" -DNDEBUG-c-Wp,-MD
	Debug	-fpic -DLINUX -O -Wall -Wno-unknown-pragmas -Wno-address -Wno-unused-but-setvariable -DPtrIntType=long -DCSREAL_IS_FLOAT -DCPU=ARMV7 -DTARGET=\"armv7Android9.0clang8.0ndkr19b\" -c-Wp,-MD
arm64Android9.0clang8.0ndkr19b	Release	-fpic -DLINUX -O -Wall -Wno-unknown-pragmas -Wno-address -Wno-unused-but-setvariable -DPtrIntType=long -DCSREAL_IS_FLOAT -DCPU=ARM64 -DTARGET=\"arm64Android9.0clang8.0ndkr19b\" -DNDEBUG-c-Wp,-MD
	Debug	-fpic -DLINUX -O -Wall -Wno-unknown-pragmas -Wno-address -Wno-unused-but-setvariable -DPtrIntType=long -DCSREAL_IS_FLOAT -DCPU=ARM64 -DTARGET=\"arm64Android9.0clang8.0ndkr19b\" -c-Wp,-MD
All supported Android architectures for Java	Release	-target 1.5 -source 1.5
	Debug	-target 1.5 -source 1.5 -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 and has been tested with both C++03 and C++11.

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**: Supported for Android 9 platforms only.
- **TCP/IPv4**: Supported.

3.4 Unsupported Features

These features are not supported on Android platforms:

- Controlling CPU Core Affinity
- Distributed Logger (supported on Android 9 only)
- Durable Writer History and Durable Reader State
- Zero Copy Transfer Over Shared Memory
- On Android 6 and higher platforms: Using DDS_WireProtocolQosPolicyAutoKind's RTPS_AUTO_ID_FROM_MAC to calculate the GUID prefix is not supported
- Internal setting of thread names at the operating-system level

3.5 Monotonic Clock Support

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

3.6 Thread Configuration

See Table 3.5 Thread Settings for Android Platforms and Table 3.6 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	
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	

3.7 Support for 'Find Package' CMake Script

The 'Find Package' CMake script is supported on Android 5.0, 5.1, and Android 9 platforms.

For information on using this script, see Building Applications using CMake, in the RTI Connext DDS Core Libraries User's Manual.

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 *Connext 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 Notes* 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 libnddstransporttlszd.a	

3.10 Libraries Required for Using RTI TCP Transport and TLS Support 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. If you are using *RTI TLS Support*, also link against the libraries in Table 3.10 Additional Libraries for using RTI TCP Transport APIs on Android Systems with TLS Enabled. 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]
Dynamic Release	libnddstransporttcp.so

[a] These 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 3.9 Additional Libraries for using RTI TCP Transport APIs on Android Systems

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

Table 3.10 Additional Libraries for using RTI TCP Transport APIs on Android Systems with TLS Enabled

Library Format	RTI TCP Transport Libraries[b]
Dynamic Release	libnddstls.so
Dynamic Debug	libnddstlsd.so
Static Release	libnddstlsz.a
Static Debug	libnddstlszd.a
OpenSSL Libraries	librtisslsupport.so

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

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

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 *Custom-supported target platform. Contact your RTI sales representative or sales@rti.com for more information.*	MPC8349	Multi 4.2.4	GHnet2 TCP/IP stack	mpc8349Inty5.0.11.mds8349
INTEGRITY 10.0.2	x86	Multi 5.0.6	GHNet IPv4 stack	pentiumInty10.0.2.pcx86[b]
INTEGRITY 11.0.4	P4080	Multi 6.1.4	GHNet2 v2	p4080Inty11.devtree-fsl-e500m-c.comp2013.5.4[c]
	Pentium class	Multi 6.1.4	GHNet2	pentiumInty11.pcx86-smp
INTEGRITY 11.4.4	x64	Multi 7.1.6	GHNet2	pentiumInty11.pcx64

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

Do not mix release and debug libraries.

See also:

---

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

[b]See 4.1 Required Patches for INTEGRITY 10.0.2 and 11.0.4 on page 30.

[c]See 4.1 Required Patches for INTEGRITY 10.0.2 and 11.0.4 on page 30.

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 or librticonnextmsgcpp2z.a	libsocket.a libnet.a libposix.a	INTEGRITY 5.0.11: RTI_INTY All others: 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 or librticonnextmsgcpp2zd.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 *cpp*.* for the Traditional C++ API or *cpp2*.* 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, Secure, Basic, or Evaluation 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
mpc8349Inty5.0.11.mds8349	Static Release	-bspname=mds8349 -prefixed_msgs --unknown_pragma_silent -G -DPtrIntType=long -DCSREAL_IS_FLOAT -DCPU=-DTARGET=”mpc8349Inty5.0.11.mds8349\” -c
	Static Debug	
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.pcx64	Static Release C	-bsp=pcx64 -prefixed_msgs -x86_64 -cpu=pentium4
	Static Release C++	-bsp=pcx64 -prefixed_msgs -x86_64 -cpu=pentium4 --c++11 exceptions
	Static Debug C	-G -bsp=pcx64 -prefixed_msgs -x86_64 -cpu=pentium4
	Static Debug C++	-G -bsp=pcx64 -prefixed_msgs -x86_64 -cpu=pentium4 --c++11 exceptions
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-T=\"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\"

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.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 all INTEGRITY platforms except INTEGRITY 5.0.11, and has been tested with C++03.

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 Unsupported Features

The RTI FlatData™ language binding is not supported on INTEGRITY 5.0.11 (mpc8349Inty5.0.11.mds8349) platforms.

The Modern C++ API for Request-Reply Communication is not supported on INTEGRITY 10.0.2 (pentiumInty10.0.2.pcx86) and INTEGRITY 5.0.11(mpc8349Inty5.0.11.mds8349) platforms. The Traditional C++ API for Request-Reply is supported.

These features are not supported on any INTEGRITY platform:

- Controlling CPU Core Affinity
- Durable Writer History and Durable Reader State
- 'Find Package' CMake script
- Monotonic clock

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	16
	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	60
	stack_size	32*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*32*1024
	cpu_list	CPU core affinity not supported
	cpu_rotation	CPU core affinity not supported
ReceiverPool threads	mask	DDS_THREAD_SETTINGS_STDIO
	priority	100
	stack_size	4*32*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
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.

4.7 Libraries Required for Using Distributed Logger

RTI Distributed Logger is supported on INTEGRITY 11.0.4 on a P4080 CPU (p4080Inty11.devtree-fsl-e500mc.comp2013.5.4) and INTEGRITY 11.4.4 on an x64 CPU (pentiumInty11.pcx64). 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)

4.8 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.

Notes:

- The RTI library in Table 4.9 Additional Libraries for Using Monitoring must appear *first* in the list of libraries to be linked.
- Automatic loading of the dynamic monitoring library through QoS is not supported.
- Memory and CPU usage is not available in monitoring data.

Table 4.9 Additional Libraries for Using Monitoring

Static Release	Static Debug
librtimonitoringz.a	librtimonitoringzd.a

[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).

4.9 Libraries Required for Zero Copy Transfer Over Shared Memory

To use the Zero Copy transfer over shared memory feature, link against the additional library in Table 4.10 Additional Libraries for Zero Copy Transfer Over Shared Memory.

Table 4.10 Additional Libraries for Zero Copy Transfer Over Shared Memory

Static Release	Static Debug
libnddsmetpz.a	libnddsmetpzd.a

4.10 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.*

4.11 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**.

4.12 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.

4.13 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. The mismatch in transport configuration between INTEGRITY and other platforms applies to *Connext DDS* 5.1.0

and higher. For more information, see the "Transport Compatibility" section in the *RTI Connext DDS Core Libraries Release Notes* for 5.3.1.

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.

4.13.1 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.

4.13.2 Using Shared Memory on INTEGRITY Systems

Connext DDS uses the single address-space POSIX library to implement the shared-memory transport on INTEGRITY 5.0 and 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.

4.13.3 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.

4.14 Using rtiddsping and rtiddsspy on INTEGRITY Systems

While the RTI libraries for INTEGRITY can be used with any BSP, providing the 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 pcx86-smp). In this case, you will need to re-integrate the *rtiddsping* and *rtiddsspy* applications. Please refer to your hardware documentation for peripheral compatibility across BSPs.

4.15 Issues with INTEGRITY Systems

4.15.1 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]

4.15.2 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.

4.15.3 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.

4.15.4 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 INtime Platforms

Table 5.1 Custom Supported INtime Platforms lists the supported INtime® platforms.

Table 5.1 Custom Supported INtime Platforms

Operating System	CPU	Compiler	RTI Architecture Abbreviation
INtime 6.3 *Custom-supported target platform. Contact your RTI sales representative or sales@rti.com for more information.*	x86	Microsoft Visual Studio 2017	i86INtime6.3VS2017

For more information on using this platform, including important prerequisites, see the RTI Connext DDS Core Libraries Getting Started Guide Addendum for INtime Systems.

Table 5.2 Building Instructions lists the compiler flags and libraries you will need to link into your application.

Table 5.3 Running Instructions shows the environment variables required to be set at run time.

Table 5.4 Library-Creation Details provides details on how these custom 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.

See also:

- 5.6 Libraries Required for Using Distributed Logger Support on page 45
- 5.7 Libraries Required for Using Monitoring on page 45
- 5.8 Libraries Required for Zero Copy Transfer over Shared Memory on page 46

Table 5.2 Building Instructions

API	Library Format	Required RTI Libraries[abc]	Required INtime Libraries	Required Compiler Flags[d]
C++ (Traditional and Modern APIs)	Static Release	nddscppz.lib or nddscpp2z.lib nddscz.lib nddscorez.lib rticonnextmsgcppz.lib or rticonnextmsgcpp2z.lib	iwin32.lib cpplib17.lib rt.lib pcibus.lib netlib.lib clib.lib vshelper.lib	/Gd /GR /EHa /MT /D "RTI_INTIME" /D "NDEBUG" /FI"windows.h" /FI"iwin32.h" /FI"winbase.h" /D "RTI_STATIC"
	Static Debug	nddscppzd.lib or nddscpp2zd.lib nddsczd.lib nddscorezd.lib rticonnextmsgcppzd.lib or rticonnextmsgcpp2zd.lib	iwin32.lib cpplib17d.lib rt.lib pcibus.lib netlib.lib clib.lib vshelper.lib	/Gd /GR /EHa /MT /D "RTI_INTIME" /D "_DEBUG" /FI"windows.h" /FI"iwin32.h" /FI"winbase.h" /D "RTI_STATIC"
	Dynamic Release	nddscpp.lib or nddscpp2.lib nddsc.lib nddscore.lib rticonnextmsgcpp.lib or rticonnextmsgcpp2.lib	iwin32.lib cpplib17.lib rt.lib pcibus.lib netlib.lib clib.lib vshelper.lib	/Gd /GR /EHa /MT /D "RTI_INTIME" /D "NDEBUG" /FI"windows.h" /FI"iwin32.h" /FI"winbase.h" /D "NDDS_DLL_VARIABLE"
	Dynamic Debug	nddscppd.lib or nddscpp2d.lib nddscd.lib nddscored.lib rticonnextmsgcppd.lib or rticonnextmsgcpp2d.lib	iwin32.lib cpplib17d.lib rt.lib pcibus.lib netlib.lib clib.lib vshelper.lib	/Gd /GR /EHa /MT /D "RTI_INTIME" /D "_DEBUG" /FI"windows.h" /FI"iwin32.h" /FI"winbase.h" /D "NDDS_DLL_VARIABLE"

[a]The C/C++ libraries are in <NDDSHOME>/lib/<architecture> (where <NDDSHOME> is where *Connext DDS* is installed, such as /home/your user name/rti_connext_dds-6.x.y).

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

[c]Choose *cpp*.* for the Traditional C++ API or *cpp2*.* for the Modern C++ API.

[d]Required INtime flags are also needed; see http://support.tenasys.com/INtimeHelp_62/Using_Develop.html#i-in-this-topic-2da985ef-15c5-40a0-9567-9902d909929f.

Table 5.2 Building Instructions

API	Library Format	Required RTI Libraries[abc]	Required INtime Libraries	Required Compiler Flags[d]
C	Static Release	nddscz.lib nddscorez.lib rticonnextmsgcz.lib	iwin32.lib rt.lib pcibus.lib netlib.lib clib.lib vshelper.lib	/Gd /GR /EHa /MT /D "RTI_INTIME" /D "NDEBUG" /FI"windows.h" /FI"iwin32.h" /FI"winbase.h" /D "RTI_STATIC"
	Static Debug	nddsczd.lib nddscorezd.lib rticonnextmsgczd.lib	iwin32.lib rt.lib pcibus.lib netlib.lib clib.lib vshelper.lib	/Gd /GR /EHa /MT /D "RTI_INTIME" /D "_DEBUG" /FI"windows.h" /FI"iwin32.h" /FI"winbase.h" /D "RTI_STATIC"
	Dynamic Release	nddsc.lib nddscore.lib rticonnextmsgc.lib	iwin32.lib rt.lib pcibus.lib netlib.lib clib.lib vshelper.lib	/Gd /GR /EHa /MT /D "RTI_INTIME" /D "NDEBUG" /FI"windows.h" /FI"iwin32.h" /FI"winbase.h" /D "NDDS_DLL_VARIABLE"
	Dynamic Debug	nddscd.lib nddscored.lib rticonnextmsgcd.lib	iwin32.lib rt.lib pcibus.lib netlib.lib clib.lib vshelper.lib	/Gd /GR /EHa /MT /D "RTI_INTIME" /D "_DEBUG" /FI"windows.h" /FI"iwin32.h" /FI"winbase.h" /D "NDDS_DLL_VARIABLE"

[a]The C/C++ libraries are in <NDDSHOME>/lib/<architecture> (where <NDDSHOME> is where *Connext DDS* is installed, such as /home/your user name/rti_connext_dds-6.x.y).

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

[c]Choose *cpp*.* for the Traditional C++ API or *cpp2*.* for the Modern C++ API.

[d]Required INtime flags are also needed; see http://support.tenasys.com/INtimeHelp_62/Using_Develop.html#i-in-this-topic-2da985ef-15c5-40a0-9567-9902d909929f.

Table 5.3 Running Instructions

RTI Architecture	Library Format (Release and Debug)	Environment Variables[a]
i86INtime6.3VS2017	Static	None required
	Dynamic	Path=%NDDSHOME%\lib\<*architecture*>; %Path%

Table 5.4 Library-Creation Details

RTI Architecture	API	Library Format	Compiler Flags Used by RTI
i86INtime6.3VS2017	C	Dynamic Release	/nologo /D "__C99__" /D "__INTIME__" /W3 /FS /WX- /FC /X /Zi /openmp- /Gy- /Zc:inline /fp:except- /Gm- /Oy- -DPtrIntType=long -DCSREAL_IS_FLOAT -DCPU=I80586 -DTARGET=\"i86INtime6.3VS2017\" /FI"windows.h" /FI"iwin32.h" /FI"winbase.h" /std:c++latest /O2 /MT /GS- -DNDEBUG
		Dynamic Debug	/nologo /D "__C99__" /D "__INTIME__" /W3 /FS /WX- /FC /X /Zi /openmp- /Gy- /Zc:inline /fp:except- /Gm- /Oy- -DPtrIntType=long -DCSREAL_IS_FLOAT -DCPU=I80586 -DTARGET=\"i86INtime6.3VS2017\" /FI"windows.h" /FI"iwin32.h" /FI"winbase.h" /std:c++latest /Od /MT /GS /D "_DEBUG"
		Static Release	/nologo /D "__C99__" /D "__INTIME__" /W3 /FS /WX- /FC /X /Zi /openmp- /Gy- /Zc:inline /fp:except- /Gm- /Oy- -DPtrIntType=long -DCSREAL_IS_FLOAT -DCPU=I80586 -DTARGET=\"i86INtime6.3VS2017\" /FI"windows.h" /FI"iwin32.h" /FI"winbase.h" /std:c++latest /O2 /MT /GS- -DNDEBUG
		Static Debug	/nologo /D "__C99__" /D "__INTIME__" /W3 /FS /WX- /FC /X /Zi /openmp- /Gy- /Zc:inline /fp:except- /Gm- /Oy- -DPtrIntType=long -DCSREAL_IS_FLOAT -DCPU=I80586 -DTARGET=\"i86INtime6.3VS2017\" /FI"windows.h" /FI"iwin32.h" /FI"winbase.h" /std:c++latest /Od /MT /GS /D "_DEBUG"

[a] %Path% represents the value of the Path variable prior to changing it to support *Connext DDS*.

Table 5.4 Library-Creation Details

RTI Architecture	API	Library Format	Compiler Flags Used by RTI
i86INtime6.3VS2017	C++	Dynamic Release	/TP /EHa /D"__INTIME_CPP17" /D"_HAS_NAMESPACE" /D"_SILENCE_ALL_CXX17_DEPRECATION_WARNINGS" /D"UNICODE" /D"_UNICODE" /D"__INTIME_CPP17_ITERATORS" /nologo /D"__C99__" /D"__INTIME__" /W3 /FS /WX- /FC /X /Zi /openmp- /Gy- /Zc:inline /fp:except- /Gm- /Oy- -DPtrIntType=long -DCSREAL_IS_FLOAT -DCPU=I80586 -DTARGET=\"i86INtime6.3VS2017\" /FI"windows.h" /FI"iwin32.h" /FI"winbase.h" /std:c++latest /O2 /MT /GS- -DNDEBUG
		Dynamic Debug	/TP /EHa /D"__INTIME_CPP17" /D"_HAS_NAMESPACE" /D"_SILENCE_ALL_CXX17_DEPRECATION_WARNINGS" /D"UNICODE" /D"_UNICODE" /D"__INTIME_CPP17_ITERATORS" /nologo /D"__C99__" /D"__INTIME__" /W3 /FS /WX- /FC /X /Zi /openmp- /Gy- /Zc:inline /fp:except- /Gm- /Oy- -DPtrIntType=long -DCSREAL_IS_FLOAT -DCPU=I80586 -DTARGET=\"i86INtime6.3VS2017\" /FI"windows.h" /FI"iwin32.h" /FI"winbase.h" /std:c++latest /Od /MT /GS /D"_DEBUG"
		Static Release	/TP /EHa /D"__INTIME_CPP17" /D"_HAS_NAMESPACE" /D"_SILENCE_ALL_CXX17_DEPRECATION_WARNINGS" /D"UNICODE" /D"_UNICODE" /D"__INTIME_CPP17_ITERATORS" /nologo /D"__C99__" /D"__INTIME__" /W3 /FS /WX- /FC /X /Zi /openmp- /Gy- /Zc:inline /fp:except- /Gm- /Oy- -DPtrIntType=long -DCSREAL_IS_FLOAT -DCPU=I80586 -DTARGET=\"i86INtime6.3VS2017\" /FI"windows.h" /FI"iwin32.h" /FI"winbase.h" /std:c++latest /O2 /MT /GS- -DNDEBUG
		Static Debug	/TP /EHa /D"__INTIME_CPP17" /D"_HAS_NAMESPACE" /D"_SILENCE_ALL_CXX17_DEPRECATION_WARNINGS" /D"UNICODE" /D"_UNICODE" /D"__INTIME_CPP17_ITERATORS" /nologo /D"__C99__" /D"__INTIME__" /W3 /FS /WX- /FC /X /Zi /openmp- /Gy- /Zc:inline /fp:except- /Gm- /Oy- -DPtrIntType=long -DCSREAL_IS_FLOAT -DCPU=I80586 -DTARGET=\"i86INtime6.3VS2017\" /FI"windows.h" /FI"iwin32.h" /FI"winbase.h" /std:c++latest /Od /MT /GS /D"_DEBUG"

5.1 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 INtime platforms.

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

5.2 Multicast Support

Multicast is supported 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.

5.3 Supported Transports

- **Shared memory:** Supported and enabled by default
- **UDPv4:** Supported and enabled by default
- **UDPv6:** Supported, with traffic class support. The transport is not enabled by default; the peers list must be modified to support IPv6.
- **TCP/IPv4:** Not supported.

5.4 Unsupported Features

These features are not supported on INtime platforms:

- Controlling CPU Core Affinity
- Durable Writer History and Durable Reader State
- 'Find Package' CMake script
- Internal setting of thread names at the operating-system level

5.5 Monotonic Clock Support

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

5.6 Libraries Required for Using Distributed Logger Support

RTI Distributed Logger is supported on INtime platforms. Table 5.5 Additional Libraries for using RTI Distributed Logger lists the additional libraries you will need in order to use *Distributed Logger*.

Table 5.5 Additional Libraries for using RTI Distributed Logger

Language	Static		Dynamic	
	Release	Debug	Release	Debug
C	rtidlcz.lib	rtidlczd.lib	rtidlc.rsl	rtidlcd.rsl
C++ (Traditional API)	rtidlcz.lib rtidlcppz.lib	rtidlczd.lib rtidlcppzd.lib	rtidlc.rsl rtidlcpp.rsl	rtidlcd.rsl rtidlcppd.rsl

5.7 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 5.6 Additional Libraries for Using Monitoring

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

5.8 Libraries Required for Zero Copy Transfer over Shared Memory

To use the Zero Copy transfer over shared memory feature, link against the additional library in Table 5.7 Additional Libraries for Zero Copy Transfer over Shared Memory.

Table 5.7 Additional Libraries for Zero Copy Transfer over Shared Memory

Static Release	Static Debug	Dynamic Release	Dynamic Debug
nddsmetpz.lib	nddsmetpzd.lib	nddsmetp.rsl	nddsmetpd.rsl

Chapter 6 iOS Platforms

Table 6.1 iOS Platforms lists the supported iOS architectures.

Table 6.1 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 6.2 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.

See also:

- 6.6 Libraries Required for Using Distributed Logger on page 52
- 6.7 Libraries Required for Using Monitoring on page 52
- 6.8 Libraries Required for Using RTI Secure WAN Transport on page 53

Table 6.2 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 or librticonnextmsgcpp2z.a	For arm64iOS8clang6.1: -arch arm64 For x86_64iOS8clang6.1: -arch x86_64	-DRTI_UNIX -DRTI_IOS
	Static Debug	libnddscppzd.a or libnddscpp2zd.a libnddsczd.a libnddscorezd.a librticonnextmsgcppzd.a or librticonnextmsgcpp2zd.a		

Universal Libraries Not Supported:

RTI does not package architecture support in universal libraries. This is done to minimize deployment size. If you prefer universal libraries, you can create them with **libtool**. For example, to create a combined library for nddscorez.a in a directory iOS8clang6.1:

```
cd $NDDSHOME/lib
mkdir iOS8clang6.1
libtool -o iOS8clang6.1/libnddscorez.a arm64iOS8clang6.1/libnddscorez.a x86_
64iOS8clang6.1/libnddscorez.a
```

Repeat for all the other *Connext DDS* libraries.

[a] Choose *cpp*.* for the Traditional C++ API or *cpp2*.* for the Modern C++ API

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

Table 6.3 Running Instructions for iOS Architectures provides details on the environment variables that must be set at run time.

Table 6.3 Running Instructions for iOS Architectures

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

Table 6.4 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 compile your application. You may find this information useful if you are involved in any in-depth debugging.

Table 6.4 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	

6.1 Supported Languages

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

6.1.1 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 and has been tested with both C++03 and C++11.

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 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.

6.3 Transports

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

6.4 Unsupported Features

These features are not supported for iOS platforms:

- Controlling CPU Core Affinity
- Durable Writer History and Durable Reader State
- 'Find Package' CMake script
- Monotonic clock
- Zero Copy Transfer Over Shared Memory
- RTI DDS Ping and Spy
- RTI Prototyper

6.5 Thread Configuration

See Table 6.5 Thread Settings for iOS Platforms and Table 6.6 Thread-Priority Definitions for iOS Platforms.

Table 6.5 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
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 6.6 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	

6.6 Libraries Required for Using Distributed Logger

RTI Distributed Logger is supported on the platforms in Table 6.1 iOS Platforms.

To use the Distributed Logger APIs, link against the additional libraries in Table 6.7 Additional Libraries for using RTI Distributed Logger .

Table 6.7 Additional Libraries for using RTI Distributed Logger

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

6.7 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 the following table must appear *first* in the list of libraries to be linked.

Table 6.8 Additional Libraries for Using Monitoring

Static Release	Static Debug
librtimonitoringz.a	librtimonitoringzd.a

6.8 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 (or if not already installed, you can find the documentation here: https://community.rti.com/documentation).

To use the Secure WAN Transport APIs, link against the additional libraries in Table 6.9 Additional Libraries for using RTI Secure WAN Transport APIs on iOS Systems. (Select the files appropriate for your chosen library format.)

Table 6.9 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 7 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 7.1 Linux Platforms on Intel CPUs below and Table 7.2 Linux Platforms on non-Intel CPUs on page 56 list the supported Linux architectures.

Table 7.1 Linux Platforms on Intel CPUs

Operating System	CPU	Compiler	RTI Architecture Abbreviation
CentOS™ 6.0, 6.2-6.4 (2.6 kernel)	x86	gcc 4.4.5	i86Linux2.6gcc4.4.5
		Java Platform, Standard Edition JDK 1.8	
	x64	gcc 4.4.5	x64Linux2.6gcc4.4.5
		Java Platform, Standard Edition JDK 1.8	
CentOS 7.0 (3.x kernel)	x86	gcc 4.8.2	i86Linux3gcc4.8.2
		Java Platform, Standard Edition JDK 1.8	
	x64	gcc 4.8.2	x64Linux3gcc4.8.2
		Java Platform, Standard Edition JDK 1.8	
Red Hat® Enterprise Linux 5.2 *Custom-supported target platform. Contact your RTI sales representative or sales@rti.com for more information.*	x86	gcc 4.2.1	i86Linux2.6gcc4.2.1
		Sun Java Platform Standard Edition JDK 1.8	

Table 7.1 Linux Platforms on Intel CPUs

Operating System	CPU	Compiler	RTI Architecture Abbreviation
Red Hat Enterprise Linux 6.0-6.5, 6.7, 6.8 (2.6 kernel)	x86	gcc 4.4.5	i86Linux2.6gcc4.4.5
		Java Platform, Standard Edition JDK 1.8	
	x64	gcc 4.4.5	x64Linux2.6gcc4.4.5
		Java Platform, Standard Edition JDK 1.8	
Red Hat Enterprise Linux 7.0, 7.3, 7.5 (3.x kernel)	x86	gcc 4.8.2	i86Linux3gcc4.8.2
		Java Platform, Standard Edition JDK 1.8	
	x64	gcc 4.8.2	x64Linux3gcc4.8.2
		Java Platform, Standard Edition JDK 1.8	
Red Hat Enterprise Linux 8.0	x64	gcc 7.3.0	x64Linux4gcc7.3.0
		Java Platform, Standard Edition JDK 1.8	
RedHawk Linux 6.0 *Custom-supported target platform. Contact your RTI sales representative or sales@rti.com for more information.*	x64	gcc 4.4.5	x64Linux2.6gcc4.4.5
RedHawk Linux 6.5 *Custom-supported target platform. Contact your RTI sales representative or sales@rti.com for more information.*	x86	gcc 4.9.2	i86RedHawk6.5gcc4.9.2
		Java Platform, Standard Edition JDK 1.8	
	x64	gcc 4.9.2	x86RedHawk6.5gcc4.9.2
		Java Platform, Standard Edition JDK 1.8	
SUSE Linux Enterprise Server 11 SP2 (3.x kernel)	x86	gcc 4.3.4	i86Linux3gcc4.3.4
		Java Platform, Standard Edition JDK 1.8	
SUSE Linux Enterprise Server 11 SP2, SP3 (2.6 kernel) SUSE Linux Enterprise Server 12 SP2 (2.6 kernel)	x64	gcc 4.3.4	x64Linux2.6gcc4.3.4
		Java Platform, Standard Edition JDK 1.8	

Table 7.1 Linux Platforms on Intel CPUs

Operating System	CPU	Compiler	RTI Architecture Abbreviation
Ubuntu® 12.04 LTS	x86	gcc 4.6.3	i86Linux3.xgcc4.4.3
		Java Platform, Standard Edition JDK 1.8	
	x64	gcc 4.6.3	x64Linux3.xgcc4.6.3
		Java Platform, Standard Edition JDK 1.8	
Ubuntu 14.04 LTS	x86	gcc 4.8.2	i86Linux3gcc4.8.2
		Java Platform, Standard Edition JDK 1.8	
	x64	gcc 4.8.2	x64Linux3gcc4.8.2
		Java Platform, Standard Edition JDK 1.8	
Ubuntu 16.04 LTS	x86	gcc 5.4.0	i86Linux3gcc5.4.0
		Java Platform, Standard Edition JDK 1.8	
	x64	gcc 5.4.0	x64Linux3gcc5.4.0
		Java Platform, Standard Edition JDK 1.8	
Ubuntu 18.04 LTS	x64	gcc 7.3.0	x64Linux4gcc7.3.0
		Java Platform, Standard Edition JDK 1.8	
Wind River® Linux 7	x64	gcc 4.9.1	x64WRLinux7gcc4.9.1

Table 7.2 Linux Platforms on non-Intel CPUs

Operating System	CPU	Compiler	RTI Architecture Abbreviation
Debian® 7 *Custom-supported target platform. Contact your RTI sales representative or sales@rti.com for more information.*	Arm v7	gcc 4.9.3	armv7aLinux3.12gcc4.9.3cortex-a9

Table 7.2 Linux Platforms on non-Intel CPUs

Operating System	CPU	Compiler	RTI Architecture Abbreviation
Freescale™ 1.4 *Custom-supported target platform. Contact your RTI sales representative or sales@rti.com for more information.*	QorIQ™ P4080/P4040/P4081	gcc 4.7.2	ppce500mcLinux3gcc4.7.2
NI Linux 3	Arm v7	gcc 4.4.1	armv7AngstromLinux3.2gcc4.4.1.cortex-a9 [a]
Raspbian Wheezy 7.0 (3.x kernel)	Arm v6	gcc 4.7.2[b] Java Platform, Standard Edition JDK 1.8	armv6vfphLinux3.xgcc4.7.2
Ubuntu 16.04 LTS	Arm v8	gcc 5.4.0 Java Platform, Standard Edition JDK 1.8	armv8Linux4.4gcc5.4.0
Wind River Linux 7.0.0.22 *Custom-supported target platform. Contact your RTI sales representative or sales@rti.com for more information.*	Arm v7	gcc 4.9.1	armv7aWRLinux7gcc4.9.1cortex-a15
Wind River Linux 8 *Custom-supported target platform. Contact your RTI sales representative or sales@rti.com for more information.*	Arm v7	gcc 5.2.0 Java Platform, Standard Edition JDK 1.8	armv7aWRLinux8gcc5.2.0
Xilinx® 14.2 *Custom-supported target platform. Contact your RTI sales representative or sales@rti.com for more information.*	Arm v7 (Zynq® Cortex A9)	gcc 4.6.1 arm-xilinx-linux-gnueabi-gcc (Sourcery CodeBench Lite 2011.09-50)	armv7Linux3.0gcc4.6.1.cortex-a9
Yocto Project® 2.2 *Custom-supported target platform. Contact your RTI sales representative or sales@rti.com for more information.*	64-bit Arm v8	gcc 6.2.0 (aarch64-linux-gnu-gcc)	armv8Linux4.9gcc6.2.0
Yocto Project 2.5 *Custom-supported target platform. Contact your RTI sales representative or sales@rti.com for more information.*	Arm v7	gcc 7.3.0	armv7Linuxgcc7.3.0

Table 7.3 Building Instructions for Linux Architectures on page 59 lists the compiler flags and libraries you will need to link into your application.

[a] These libraries require a hardware FPU in the processor and are compatible with systems that have soft-float libc. See platform notes for compiler flag details.

[b] Requires Linaro Gnueabihf Cross Compiler.

See also:

- 7.10 Libraries Required for Using Distributed Logger on page 69
- 7.11 Libraries Required for Using Monitoring on page 70
- 7.12 Libraries Required for Using RTI Secure WAN Transport APIs on page 70
- 7.13 Libraries Required for Using RTI TCP Transport and TLS Support APIs on page 71
- 7.14 Libraries Required for Zero Copy Transfer Over Shared Memory on page 72

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 7.3 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 librticonnextmsgcppz.a or librticonnextmsgcpp2z.a	For x64Linux4gcc7.3.0, i86Linux2.6gcc4.4.5 and x64Linux2.6gcc4.4.5: -ldl -lm -lpthread -lrt For all other architectures: -ldl -lnsl -lm -lpthread -lrt For Yocto Project, also add: -L/usr/lib/nptl	For 64-bit architectures (except Yocto Project): -DRTI_UNIX -m64 For 32-bit architectures: -DRTI_UNIX -m32 For Ubuntu: For dynamic release and dynamic debug libraries, also add: -Wl,--no-as-needed For Wind River Linux 7/8 on Arm v7, Xilinx: -DRTI_UNIX For Yocto Project: -DRTI_UNIX -m32 --sysroot =SYSROOT
	Static Debug	libnddscppzd.a or libnddscpp2zd.a libnddsczd.a libnddscorezd.a librticonnextmsgcppzd.a or librticonnextmsgcpp2zd.a		
	Dynamic Release	libnddscpp.so or libnddscpp2.so libnddsc.so libnddscore.so librticonnextmsgcpp.so or librticonnextmsgcpp2.so		
	Dynamic Debug	libnddscppd.so or libnddscpp2d.so libnddscd.so libnddscored.so librticonnextmsgcppd.so or librticonnextmsgcpp2d.so		

[a]Choose *cpp*.* for the Traditional C++ API or *cpp2*.* 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, Secure, Basic, or Evaluation package type. It is not provided with the Core package type.

Table 7.3 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	For x64Linux4gcc7.3.0, i86Linux2.6gcc4.4.5 and x64Linux2.6gcc4.4.5: -ldl -lm -lpthread -lrt For all other architectures: -ldl -lnsl -lm -lpthread -lrt For Yocto Project, also add: -L/usr/lib/nptl	For 64-bit architectures (except Yocto Project): -DRTI_UNIX -m64 For 32-bit architectures: -DRTI_UNIX -m32 For Ubuntu: For dynamic release and dynamic debug libraries, also add: -Wl,--no-as-needed For Wind River Linux 7/8 on Arm v7, Xilinx: -DRTI_UNIX For Yocto Project: -DRTI_UNIX -m32 --sysroot =SYSROOT
	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 7.4 Running Instructions for Linux Architectures on the next page 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 *cpp*.* for the Traditional C++ API or *cpp2*.* 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, Secure, Basic, or Evaluation package type. It is not provided with the Core package type.

Table 7.4 Running Instructions for Linux Architectures

RTI Architecture	Library Format	Environment Variables
All supported Linux 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 supported Linux architectures when not using Java	Static (Release & Debug)	None required
	Dynamic (Release & Debug)	LD_LIBRARY_PATH= ${NDDSHOME}/lib/<*architecture*>: ${LD_LIBRARY_PATH}

Table 7.5 Library-Creation Details for Linux Architectures below 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 7.5 Library-Creation Details for Linux Architectures

RTI Architecture	Library Format (Static & Dynamic)	Compiler Flags Used by RTI
armv6vfphLinux3.x gcc4.7.2	Release	-fPIC -DLINUX -O -Wall -Wno-unknown-pragmas -DPtrIntType=long -DCSREAL_IS_FLOAT -DCPU-U=ARMV6 -DTARGET=\"armv6vfphLinux3.xgcc4.7.2\" -DNDEBUG -c -Wp,-MD
	Debug	-fPIC -g -DLINUX -O -Wall -Wno-unknown-pragmas -DPtrIntType=long -DCSREAL_IS_FLOAT -DCPUU-U=ARMV6 -DTARGET=\"armv6vfphLinux3.xgcc4.7.2\" -c -Wp,-MD
armv7Angstrom Linux3.2 gcc4.4.1.cortex-a9	Release	-fpic -DLINUX -O -Wall -Wno-unknown-pragmas -DPtrIntType=long -DCSREAL_IS_FLOAT -DCPU-U=cortex-a9 -DTARGET=\"armv7AngstromLinux3.2gcc4.4.1.cortex-a9\" -DNDEBUG -c -Wp,-MD
	Debug	-fpic -DLINUX -g -Wall -Wno-unknown-pragmas -DPtrIntType=long -DCSREAL_IS_FLOAT -DCPU-U=cortex-a9 -DTARGET=\"armv7AngstromLinux3.2gcc4.4.1.cortex-a9\" -c -Wp,-MD
armv7aLinux3.12 gcc4.9.3cortex-a9	Release	-fpic -DLINUX -march=armv7-a -mtune=cortex-a9 -mfloat-abi=hard -mfpu=vfpv3-d16 -mthumb -mtls-dialect=gnu -DPtrIntType=long -DCSREAL_IS_FLOAT -DCPU=cortex-a9 -DRTI_LINUX -DNDEBUG -Wp,-MD
	Debug	-fpic -DLINUX -march=armv7-a -mtune=cortex-a9 -mfloat-abi=hard -mfpu=vfpv3-d16 -mthumb -mtls-dialect=gnu -DPtrIntType=long -DCSREAL_IS_FLOAT -DCPU=cortex-a9 -DRTI_LINUX -Wp,-MD
armv7aWRLinux7 gcc4.9.1cortex-a15	Release	-march=armv7-a -mfloat-abi=hard -mfpu=neon -marm -mthumb-interwork -mtune=cortex-a15 -fPIC -DPtrIntType=long -DNDEBUG
	Debug	-march=armv7-a -mfloat-abi=hard -mfpu=neon -marm -mthumb-interwork -mtune=cortex-a15 -fPIC -DPtrIntType=long -g

Table 7.5 Library-Creation Details for Linux Architectures

RTI Architecture	Library Format (Static & Dynamic)	Compiler Flags Used by RTI
armv7aWRLinux8 gcc5.2.0	Release	-march=armv7-a -marm -mfpu=neon -mfloat-abi=hard -mtune=cortex-a7 -fPIC -DLINUX -O -Wall -Wno-unknown-pragmas -DPtrIntType=long -DCSREAL_IS_FLOAT -DCPU=cortex-a7 -DNDEBUG -c -Wp,-MD
	Debug	-g -march=armv7-a -marm -mfpu=neon -mfloat-abi=hard -mtune=cortex-a7 -fPIC -DLINUX -O -Wall -Wno-unknown-pragmas -DPtrIntType=long -DCSREAL_IS_FLOAT -DCPU=cortex-a7 -c -Wp,-MD
armv7Linux3.0 gcc4.6.1.cortex-a9	Release	-mcpu=cortex-a9 -march=armv7-a -mlong-calls -fpic -DLINUX -O -Wall -Wno-unknown-pragmas -DPtrIntType=long -DCSREAL_IS_FLOAT -DCPU=cortex-a9 -DTARGET-T=\"armv7Linux3.0gcc4.6.1.cortex-a9\" -DNDEBUG -c -Wp,-MD
	Debug	-mcpu=cortex-a9 -march=armv7-a -mlong-calls -fpic -DLINUX -g -Wall -Wno-unknown-pragmas -DPtrIntType=long -DCSREAL_IS_FLOAT -DCPU=cortex-a9 -DTARGET-T=\"armv7Linux3.0gcc4.6.1.cortex-a9\" -c -Wp,-MD
armv7Linuxgcc7.3.0	Release	-fPIC -DLINUX -O -Wall -Wno-unknown-pragmas -DPtrIntType=long -DCSREAL_IS_FLOAT -DCPU-U=ARMV7 -DTARGET=\"armv7Linuxgcc7.3.0\" -DNDEBUG -c -Wp,-MD
	Debug	-g -fPIC -DLINUX -O -Wall -Wno-unknown-pragmas -DPtrIntType=long -DCSREAL_IS_FLOAT -DCPU-U=ARMV7 -DTARGET=\"armv7Linuxgcc7.3.0\" -c -Wp,-MD
armv8Linux4.9gcc6.2.0	Release	-mabi=lp64 -fPIC -DLINUX -O -Wall -Wno-unknown-pragmas -DPtrIntType=long -DCSREAL_IS_FLOAT -DCPU=cortex-a57 -DTARGET=\"armv8Linux4.9gcc6.2.0\" -DNDEBUG -c -Wp,-MD
	Debug	-mabi=lp64 -fPIC -DLINUX -g -Wall -Wno-unknown-pragmas -DPtrIntType=long -DCSREAL_IS_FLOAT -DCPU=cortex-a57 -DTARGET=\"armv8Linux4.9gcc6.2.0\" -c -Wp,-MD
armv8Linux4.4gcc5.4.0	Release	-fPIC -DLINUX -O -Wall -Wno-unknown-pragmas -DPtrIntType=long -DCSREAL_IS_FLOAT -DCPU-U=ARMV8 -DTARGET=\"armv8Linux4.4gcc5.4.0\" -DNDEBUG -c -Wp,-MD
	Debug	-g -fPIC -DLINUX -O -Wall -Wno-unknown-pragmas -DPtrIntType=long -DCSREAL_IS_FLOAT -DCPU-U=ARMV8 -DTARGET=\"armv8Linux4.4gcc5.4.0\" -c -Wp,-MD
i86Linux2.6gcc4.2.1	Release	-fPIC -DLINUX -O -Wall -Wno-unknown-pragmas -DPtrIntType=long -DCSREAL_IS_FLOAT -DCPU-U=I80586 -DTARGET=\"i86Linux2.6gcc4.2.1\" -fmessage-length=0 -DNDEBUG -c -Wp,-MD
	Debug	-fPIC -DLINUX -g -Wall -Wno-unknown-pragmas -DPtrIntType=long -DCSREAL_IS_FLOAT -DCPU-U=I80586 -DTARGET=\"i86Linux2.6gcc4.2.1\" -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-U=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-U=I80586 -DTARGET=\"i86Linux3gcc4.3.4\" -fmessage-length=0 -c-Wp,-MD

Table 7.5 Library-Creation Details for Linux Architectures

RTI Architecture	Library Format (Static & Dynamic)	Compiler Flags Used by RTI
i86Linux3gcc4.8.2	Release	-m32 -fPIC -O -Wall -Wno-unknown-pragmas -DPtrIntType=long -DCSREAL_IS_FLOAT -DCPU=i686 -DTARGET=\"i86Linux3gcc4.8.2\" -DNDEBUG -c -Wp,-MD
	Debug	-g -m32 -fPIC -O -Wall -Wno-unknown-pragmas -DPtrIntType=long -DCSREAL_IS_FLOAT -DCPU=i686 -DTARGET=\"i86Linux3gcc4.8.2\" -DDEBUG -c -Wp,-MD
i86Linux3gcc5.4.0	Release	-m32 -fPIC -Wall -Wno-unknown-pragmas -Werror=implicit-function-declaration -DPtrIntType=long -DCSREAL_IS_FLOAT -DCPU=i686 -DTARGET=\"i86Linux3gcc5.4.0\" -DNDEBUG
	Debug	-m32 -fPIC -g -Wall -Wno-unknown-pragmas -Werror=implicit-function-declaration -DPtrIntType=long -DCSREAL_IS_FLOAT -DCPU=i686 -DTARGET=\"i86Linux3gcc5.4.0\"
i86RedHawk6.5gcc4.9.2	Release	-m32 -fPIC -O -Wall -Wno-unknown-pragmas -DRTS_UNIX -DPtrIntType=long -DCSREAL_IS_FLOAT -DCPU=I80586 -DTARGET=\\"i86RedHawk6.5gcc4.9.2\\" -DNDEBUG -c -Wp,-MD
	Debug	-m32 -fPIC -g -Wall -Wno-unknown-pragmas -DRTS_UNIX -DPtrIntType=long -DCSREAL_IS_FLOAT -DCPU=I80586 -DTARGET=\\"i86RedHawk6.5gcc4.9.2\\" -c -Wp,-MD
ppce500mcLinux3 gcc4.7.2	Release	-m32 -mhard-float -mcpu=e500mc -O2 -pipe -feliminate-unused-debug-types -fPIC -DPOWERPC -O -Wall -Wno-unknown-pragmas -DPtrIntType=long -DCSREAL_IS_FLOAT -DCPU=e500mc -DTARGET=\"ppce500mcLinux3gcc4.7.2\" -DNDEBUG -c -Wp,-MD
	Debug	-m32 -mhard-float -mcpu=e500mc -O2 -pipe -feliminate-unused-debug-types -fPIC -DPOWERPC -O -Wall -Wno-unknown-pragmas -DPtrIntType=long -DCSREAL_IS_FLOAT -DCPU=e500mc -DTARGET-T=\"ppce500mcLinux3gcc4.7.2\" -g -c -Wp,-MD
x64Linux2.6gcc4.3.4	Release	-m64 -fPIC -DLINUX -O -Wall -Wno-unknown-pragmas -DPtrIntType=long -DCSREAL_IS_FLOAT -DCPU-U=AMD64 -DTARGET=\"x64Linux2.6gcc\" -c -Wp,-MD
	Debug	-m64 -fPIC -DLINUX -g -Wall -Wno-unknown-pragmas -DPtrIntType=long -DCSREAL_IS_FLOAT -DCPU-U=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 -O -Wall -Wno-unknown-pragmas -DPtrIntType=long -DCSREAL_IS_FLOAT -DCPU=i686 -DTARGET=\"x64Linux3gcc4.8.2\" -DNDEBUG -c -Wp,-MD
	Debug	-g -m64 -fPIC -O -Wall -Wno-unknown-pragmas -DPtrIntType=long -DCSREAL_IS_FLOAT -DCPU=i686 -DTARGET=\"x64Linux3gcc4.8.2\" -DDEBUG -c -Wp,-MD

Table 7.5 Library-Creation Details for Linux Architectures

RTI Architecture	Library Format (Static & Dynamic)	Compiler Flags Used by RTI
x64Linux3gcc5.4.0	Release	-m64 -fPIC -O -Wall -Wno-unknown-pragmas -Werror=implicit-function-declaration-DPtrIntType=long -DCSREAL_IS_FLOAT -DCPU=AMD64 -DTARGET=\"x64Linux3gcc5.4.0\" -DNDEBUG
	Debug	-m64 -fPIC -g -O -Wall -Wno-unknown-pragmas -Werror=implicit-function-declaration -DPtrIntType=long -DCSREAL_IS_FLOAT -DCPU=AMD64 -DTARGET=\"x64Linux3gcc5.4.0\"
x64Linux4gcc7.3.0	Release	-fPIC -DRTI_CPU_AFFINITY -O -DPtrIntType=long -DCSREAL_IS_FLOAT -DCPU=AMD64 -DTARGET-T=\"x64Linux4gcc7.3.0\" -DNDEBUG
	Debug	-fPIC -DRTI_CPU_AFFINITY -g -Wextra -DPtrIntType=long -DCSREAL_IS_FLOAT -DCPU=AMD64 -DTARGET=\"x64Linux4gcc7.3.0\"
x64RedHawk6.5gcc4.9.2	Release	-fPIC -DLINUX -O -Wall -Wno-unknown-pragmas -DRTS_UNIX -DPtrIntType=long -DCSREAL_IS_FLOAT -DCPU=X86_64 -DTARGET=\\"x64RedHawk6.5gcc4.9.2\\" -DNDEBUG -c -Wp,-MD
	Debug	-fPIC -DLINUX -g -Wall -Wno-unknown-pragmas -DRTS_UNIX -DPtrIntType=long -DCSREAL_IS_FLOAT -DCPU=X86_64 -DTARGET=\\"x64RedHawk6.5gcc4.9.2\\" -c -Wp,-MD
x64WRLinux7gcc4.9.1	Release	-fPIC -m64 -march=x86-64 -mtune=generic -fPIC -DLINUX -O -Wall -Wno-unknown-pragmas -DPtrIntType=long -DCSREAL_IS_FLOAT -DCPU=AMD64 -DTARGET=\"x64WRLinux7gcc4.9.1\" -DNDEBUG -c -Wp,-MD
	Debug	-g -fPIC -m64 -march=x86-64 -mtune=generic -fPIC -DLINUX -O -Wall -Wno-unknown-pragmas -DPtrIntType=long -DCSREAL_IS_FLOAT -DCPU=AMD64 -DTARGET=\"x64WRLinux7gcc4.9.1\" -c -Wp,-MD
All supported Linux architectures for Java	Dynamic Release	-target 1.5 -source 1.5
	Dynamic Debug	-target 1.5 -source 1.5 -g

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 available for all the platforms in Table 7.1 Linux Platforms on Intel CPUs on page 54 and Table 7.2 Linux Platforms on non-Intel CPUs on page 56.

Some platforms have been tested only on C++03, whereas others have been tested with both C++03 and C++11. C++03 is typically supported with gcc3.4.2 and above. C++11 is typically supported with gcc4.7.2 and above.

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 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.

7.3 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 and NI Linux 3.

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

  Traffic Class support is only provided on architectures with gcc 4.1.0 or later that support the UDPv6 transport.
- **TCP/IPv4:** Supported for all Linux platforms *except* Freescale, RedHawk 6.0, Red Hat Enterprise Linux 5.2, Wind River Linux 7/8 on Arm v7, and Yocto Project. This is *not* a built-in transport.

7.3.1 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*.

7.4 Unsupported Features

Setting thread names at the operating-system level is not supported on the following architectures:

- i86Linux2.6gcc4.2.1
- i86Linux3gcc4.3.4
- i86Linux2.6gcc4.4.5

- x64Linux2.6gcc4.3.4
- x64Linux2.6gcc4.4.5
- armv6vfphLinux3.xgcc4.7.2
- armv7AngstromLinux3.2gcc4.4.1.cortex-a9
- armv7Linux3.0gcc4.6.1.cortex-a9

7.5 Monotonic Clock Support

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

7.6 Thread Configuration

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

See also: Table 7.7 Thread-Priority Definitions for Linux Platforms on the next page and Table 7.8 Thread Kinds for Linux Platforms on the next page.

7.6.1 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 available on all supported Linux/SUSE platforms.

Table 7.6 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)
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)

Table 7.6 Thread Settings for Linux 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	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 7.7 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 7.8 Thread Kinds for Linux Platforms

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

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

Table 7.8 Thread Kinds for Linux Platforms

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

7.6.2 Using REALTIME_PRIORITY

If the **mask** field includes DDS_THREAD_SETTINGS_REALTIME_PRIORITY, a value must also be explicitly specified for the "priority" field in the QoS. (This is because using DDS_THREAD_SETTINGS_REALTIME_PRIORITY changes the scheduler used by Linux for the thread to SCHED_FIFO. If the **priority** field is not explicitly set, it will default to a value of 0, but this is an invalid value for a priority when using SCHED_FIFO.) Note that running with REALTIME_PRIORITY requires the appropriate privileges: the process will need to be run with root privileges on Linux in order to set the scheduler.

7.7 Durable Writer History and Durable Reader State Features

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

- Ubuntu 12.04 LTS (i86Linux2.6gcc4.6.3, x64Linux2.6gcc4.6.3)

To use the Durable Writer History and Durable Reader State features, you must install a relational database. Only MySQL® is supported.

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

7.8 Support for 'Find Package' CMake Script

The 'Find Package' CMake script is supported on the following platforms:

- Linux platforms on Intel CPUs (see Table 7.1 Linux Platforms on Intel CPUs on page 54)
- Raspbian Wheezy 7.0 (armv6vfphLinux3.xgcc4.7.2)

For information on using this script, see Building Applications using CMake, in the RTI Connext DDS Core Libraries User's Manual.

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

7.9 Backtrace Support

Starting with Ubuntu 16.10, GCC by default has been modified to compile programs with PIE (position independent executable) support on the amd64 and ppc64el architectures. Therefore, RTI has compiled the following architecture using "-no-pie" as the linker option, in debug mode, to display the backtrace information; to display backtrace information on this architecture, you must also disable PIE by using the "-no-pie" linker option in your application:

- Ubuntu 18.04 LTS on x64 CPU (x64Linux4gcc7.3.0)

On Raspberry Pi (Raspbian Wheezy) architectures, the backtrace returns 0 frames. To solve this problem, RTI has compiled the following architectures with the "-funwind-tables" option to display the backtrace information, and you must do the same:

- Raspbian Wheezy 7.0 (3.x kernel) (armv6vfphLinux3.xgcc4.7.2)
- Debian 7 (armv7aLinux3.12gcc4.9.3cortex-a9)
- Wind River Linux 7.0.0.22 (armv7aWRLinux7gcc4.9.1cortex-a15)
- Wind River Linux 8 (armv7aWRLinux8gcc5.2.0)
- Ubuntu 16.04 LTS (armv8Linux4.4gcc5.4.0)
- Yocto Project 2.2 (armv8Linux4.9gcc6.2.0)
- Yocto Project 2.5 (armv7Linuxgcc7.3.0)

This compiler option creates a table that allows the program to walk back the function call stack from a given execution point.

See Logging a Backtrace for Failures, in the RTI Connext DDS Core Libraries User's Manual.

7.10 Libraries Required for Using Distributed Logger

RTI Distributed Logger is supported on all the platforms in Table 7.1 Linux Platforms on Intel CPUs on page 54 and Table 7.2 Linux Platforms on non-Intel CPUs on page 56 *except* RedHawk Linux 6.0 and Red Hat Enterprise Linux 5.2.

To use the Distributed Logger APIs, link against the additional libraries in Table 7.9 Additional Libraries for using RTI Distributed Logger . (Select the files appropriate for your chosen library format.)

Table 7.9 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
Java	N/A	N/A	distlog.jar distlogdatamodel.jar	distlogd.jar distlogdatamodeld.jar

7.11 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 in Table 7.10 Additional Libraries for Using Monitoring below must appear *first* in the list of libraries to be linked.

Table 7.10 Additional Libraries for Using Monitoring

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

7.12 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 specific architectures. See the RTI Secure WAN Transport Release Notes and RTI Secure WAN Transport Installation Guide (or if it is not already installed, you can find the documentation here: https://community.rti.com/documentation).

To use the *Secure WAN Transport* APIs, link against the additional libraries in Table 7.11 Additional Libraries for using RTI Secure WAN Transport APIs on UNIX-Based Systems below. Select the files appropriate for your chosen library format.

Table 7.11 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	

7.13 Libraries Required for Using RTI TCP Transport and TLS Support APIs

To use the TCP Transport APIs, link against the additional libraries in Table 7.12 Additional Libraries for using RTI TCP Transport APIs on UNIX-Based Systems on the next page.

If you are using *RTI TLS Support*, see Table 7.13 Additional Libraries for using RTI TCP Transport APIs on UNIX-Based Systems with TLS Enabled on the next page. Select the files appropriate for your chosen library format.

RTI TLS Support is an optional product for use with the TCP transport that is included with *RTI Connext® DDS*. If you choose to use *TLS Support*, it must be installed on top of a *Connext DDS* installation with the same version number; it can only be used on architectures that support TCP transport (see the RTI TLS Support Release Notes).

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

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

Table 7.12 Additional Libraries for using RTI TCP Transport APIs on UNIX-Based 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 7.13 Additional Libraries for using RTI TCP Transport APIs on UNIX-Based 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

7.14 Libraries Required for Zero Copy Transfer Over Shared Memory

To use the Zero Copy transfer over shared memory feature, link against the additional library in Table 7.14 Additional Libraries for Zero Copy Transfer Over Shared Memory below.

Table 7.14 Additional Libraries for Zero Copy Transfer Over Shared Memory

Static Release	Static Debug	Dynamic Release	Dynamic Debug
libnddsmetpz.a	libnddsmetpzd.a	libnddsmetp.so	libnddsmetpd.so

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

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

Chapter 8 LynxOS Platforms

Table 8.1 Supported LynxOS Platforms lists the architectures supported on LynxOS® operating systems.

Table 8.1 Supported LynxOS Platforms

Operating System	CPU	Compiler	RTI Architecture
LynxOS 5.0	PPC 74xx (such as 7410)	gcc 3.4.3	ppc7400Lynx5.0.0gcc3.4.3

Table 8.2 Building Instructions for LynxOS Architectures and Table 8.3 Building Instructions for LynxOS Architectures list the compiler flags and libraries you will need to link into your application.

See also:

- 8.6 Libraries Required for Using Monitoring on page 79
- 8.7 Libraries Required for Zero Copy Transfer Over Shared Memory on page 79

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

Table 8.5 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 8.2 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 or librticonnextmsgcpp2z.a
	Static Debug	libnddscppzd.a or libnddscpp2zd.a libnddsczd.a libnddscorezd.a librticonnextmsgcppzd.a or librticonnextmsgcpp2zd.a
	Dynamic Release	libnddscpp.so or libnddscpp2.so libnddsc.so libnddscore.so librticonnextmsgcpp.so or librticonnextmsgcpp2.so
	Dynamic Debug	libnddscppd.so or libnddscpp2d.so libnddscd.so libnddscored.so librticonnextmsgcppd.so or librticonnextmsgcpp2d.so

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

[b]Choose *cpp*.* for the Traditional C++ API or *cpp2*.* 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 1.1 Paths Mentioned in Documentation on page 4.

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

Table 8.2 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 8.3 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
	ppc7400Lynx5.0.0gcc3.4.3		For ppc7400Lynx5.0.0gcc3.4.3, also add: -RTI_LYNX500

Table 8.4 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}

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

[b]Choose *cpp*.* for the Traditional C++ API or *cpp2*.* 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 1.1 Paths Mentioned in Documentation on page 4.

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

Table 8.5 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

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 only available for the LynxOS 5.0 platform and has been tested only with C++03.

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

8.2 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.)

8.3 Supported Transports

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

8.3.1 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*.

8.4 Unsupported Features

These features are not supported on LynxOS platforms:

- Controlling CPU Core Affinity
- Distributed Logger
- Durable Writer History and Durable Reader State
- 'Find Package' CMake script
- Monotonic clock
- Internal setting of thread names at the operating-system level

8.5 Thread Configuration

See Table 8.6 Thread Settings for LynxOS Platforms and Table 8.7 Thread-Priority Definitions for LynxOS Platforms.

Table 8.6 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	64*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	64*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*64*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*64*1024
	cpu_list	CPU core affinity not supported
	cpu_rotation	CPU core affinity not supported

Table 8.7 Thread-Priority Definitions for LynxOS Platforms

Thread-Priority Definition	Operating-System Priority
THREAD_PRIORITY_DEFAULT	17
THREAD_PRIORITY_HIGH	32

Table 8.7 Thread-Priority Definitions for LynxOS Platforms

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

8.6 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.

Notes:

- Automatic loading of the dynamic monitoring library through QoS is not supported.
- Memory and CPU usage is not available in monitoring data.
- If you plan to use *static* libraries, the RTI library in 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.so	librtimonitoringd.so

8.7 Libraries Required for Zero Copy Transfer Over Shared Memory

To use the Zero Copy transfer over shared memory feature, link against the additional library in Table 8.9 Additional Libraries for Zero Copy Transfer Over Shared Memory.

Table 8.9 Additional Libraries for Zero Copy Transfer Over Shared Memory

Static Release	Static Debug	Dynamic Release	Dynamic Debug
libnddsmetpz.a	libnddsmetpzd.a	libnddsmetp.so	libnddsmetpd.so

8.8 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 9 macOS Platforms

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

Table 9.1 Supported macOS Platforms

Operating System	CPU	Compiler	RTI Architecture Abbreviation
macOS 10.12	x64	clang 8.0	x64Darwin16clang8.0
		Java Platform, Standard Edition JDK 1.8	
macOS 10.13, 10.14	x64	clang 9.0	x64Darwin17clang9.0
		Java Platform, Standard Edition JDK 1.8	

Table 9.2 Building Instructions for macOS Architectures lists the compiler flags and libraries you will need to link into your application.

Other libraries you may need are described here:

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 macOS Architectures provides details on the environment variables that must be set at run time for a macOS architecture.

Table 9.4 Library-Creation Details for macOS 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 9.2 Building Instructions for macOS 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 or librticonnextmsgcpp2z.a	-ldl -lm -lpthread	-dynamic -lpthread -lc -single_module -DRTI_UNIX -DRTI_DARWIN -DRTI_64BIT
	Static Debug	libnddscppzd.a or libnddscpp2zd.a libnddsczd.a libnddscorezd.a librticonnextmsgcppzd.a or librticonnextmsgcpp2zd.a		
	Dynamic Release	libnddscpp.dylib or libnddscpp2.dylib libnddsc.dylib libnddscore.dylib librticonnextmsgcpp.dylib or librticonnextmsgcpp2.dylib		
	Dynamic Debug	libnddscppd.dylib or libnddscpp2d.dylib libnddscd.dylib libnddscored.dylib librticonnextmsgcppd.dylib or librticonnextmsgcpp2d.dylib		

[a]Choose *cpp*.* for the Traditional C++ API or *cpp2*.* 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 1.1 Paths Mentioned in Documentation on page 4

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

Table 9.2 Building Instructions for macOS 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_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 *cpp*.* for the Traditional C++ API or *cpp2*.* 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 1.1 Paths Mentioned in Documentation on page 4

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

Table 9.3 Running Instructions for macOS Architectures

RTI Architecture	Library Format (Release & Debug)	Required Environment Variables[a]
x64Darwin16clang8.0	Static	None required
	Dynamic	DYLD_LIBRARY_PATH=${NDDSHOME}/lib/x64Darwin16clang8.0:${DYLD_LIBRARY_PATH}
x64Darwin16clang8.0 for Java	N/A	DYLD_LIBRARY_PATH=${NDDSHOME}/lib/x64Darwin16clang8.0:${DYLD_LIBRARY_PATH}
x64Darwin17clang9.0	Static	None required
	Dynamic	DYLD_LIBRARY_PATH=${NDDSHOME}/lib/x64Darwin17clang9.0:${DYLD_LIBRARY_PATH}
x64Darwin17clang9.0 for Java	N/A	DYLD_LIBRARY_PATH=${NDDSHOME}/lib/x64Darwin17clang9.0:${DYLD_LIBRARY_PATH}

Table 9.4 Library-Creation Details for macOS Architectures

RTI Architecture	Library Format (Static & Dynamic)	Compiler Flags Used by RTI
x64Darwin16clang8.0	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
x64Darwin16clang8.0 for Java	Release	-target 1.5 -source 1.5
	Debug	-target 1.5 -source 1.5 -g

[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 9.4 Library-Creation Details for macOS Architectures

RTI Architecture	Library Format (Static & Dynamic)	Compiler Flags Used by RTI
x64Darwin17clang9.0	Release	-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 -DTARGET=\"x64Darwin17clang9.0\" -DNDEBUG
	Debug	-x c -arch x86_64 -mtune=core2 -Wno-trigraphs -fpascal-strings -fasm-blocks -g -O -Wall -Wno-unknown-pragmas -DPtrIntType=long -DCSREAL_IS_FLOAT -DCPU=AMD64 -DTARGET=\"x64Darwin17clang9.0\"
x64Darwin17clang9.0 for Java	Release	-target 1.5 -source 1.5
	Debug	-target 1.5 -source 1.5 -g

9.1 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 macOS platforms and has been tested with both C++03 and C++11.

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

9.2 Multicast Support

Multicast is supported on macOS 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.

9.3 Supported Transports

- **Shared memory:** Supported and enabled by default.
- **UDPv4:** Supported and enabled by default.
- **UDPv6:** Supported on macOS 10.13, 10.14 (x64Darwin17clang9.0).
- **TCP/IPv4:** Supported.

9.4 Unsupported Features

These features are not supported on macOS platforms:

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

9.5 System Integrity Protection (SIP)

A feature called System Integrity Protection (SIP) was introduced in macOS 10.11. If enabled, this feature strips out the environment variable DYLD_LIBRARY_PATH, which is used to specify the location of shared libraries for a program. For more details, see https://support.apple.com/en-us/HT204899.

9.5.1 SIP and Java Applications

If you run *Connext DDS* applications using a Java Runtime Environment located under one of the paths protected by SIP (e.g., **/usr/bin**) and rely on the DYLD_LIBRARY_PATH environment variable to set the path to the *Connext DDS* run-time libraries (or any other third party run-time libraries, such as OpenSSL), Java will fail to load them with an error message such as:

```
The library libnddsjava.dylib could not be loaded by your operating system
```

To overcome this limitation, when running Java applications on macOS systems, you must use the **java.library.path** variable instead of the DYLD_LIBRARY_PATH environment variable to indicate the path to the *Connext DDS* libraries. This is automatically performed by the scripts to run applications generated by the *RTI Code Generator*. However, if you are manually running your *Connext DDS* application using the Java Runtime Environment, or you are writing our own scripts to run your Java application, you can indicate it as follows:

```
java -Djava.library.path="<installation_dir>/lib/<architecture>" -classpath .:"<installation_
dir>/lib/java/nddsjava.jar" <your_class>
```

Additionally, some *Connext DDS* applications may need to dynamically load functionality that is implemented in separate libraries (e.g., for the RTI Monitoring Library or transport plugins such as *RTI Secure WAN Transport* or *RTI TLS Support*). In that case, specifying the path to the **lib** directory using **java.library.path** is not sufficient, because the path to those libraries is not exposed to the underlying *Connext DDS* infrastructure.

To work around this limitation, you must provide the full path and extension of the dynamic libraries that are loaded at run time. In the case of the RTI Monitoring Library, this implies adding the following to your XML configuration file:

```
<participant_qos>
    <property>
        <value>
            <element>
                <name>rti.monitor.library</name>
                <value>/full-path-to-librtimonitoring.dylib</value>
            </element>
            <!-- ... -->
```

```
        </value>
    </property>
</participant_qos>
```

Likewise, for transport plugins that are loaded dynamically (e.g., the TCP transport plugin), you must add the full path to the XML configuration file:

```
<participant_qos>
    <property>
        <!-- ... -->
        <value>
            <element>
                <name>dds.transport.TCPv4.tcp1.library</name>
                <value>/full-path-to-libnddstransporttcp.dylib</value>
            </element>
            <!-- ... -->
        </value>
    </property>
</participant_qos>
```

For more on transport plugins, see 9.11 Libraries Required for Using RTI Secure WAN Transport APIs on page 92 and 9.12 Libraries Required for Using RTI TCP Transport APIs on page 92.

9.5.2 SIP and Connext Tools, Infrastructure Services, and Utilities

The SIP feature also makes it impossible for the scripts under **<installation_dir>/bin** to pick up the value of the DYLD_LIBRARY_PATH environment variable at run time. To workaround this issue, *Connext DDS* tools, infrastructure services, and utilities rely on 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) to your library path, you can export the RTI_LD_LIBRARY_PATH environment variable and run *Routing Service* as follows:

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

9.6 Thread Configuration

See Table 9.5 Thread Settings for macOS Platforms and Table 9.6 Thread-Priority Definitions for macOS Platforms.

Table 9.5 Thread Settings for macOS 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 9.6 Thread-Priority Definitions for macOS 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	

9.7 Support for 'Find Package' CMake Script

The 'Find Package' CMake script is supported on all macOS platforms in Table 9.1 Supported macOS Platforms.

For information on using this script, see Building Applications using CMake, in the RTI Connext DDS Core Libraries User's Manual.

9.8 Resolving NDDSUtility_sleep() Issues

When running on a macOS 10.9 (or higher) system, you may experience timing issues in your calls to **NDDSUtility_sleep()**. If you request to sleep for a small enough time period, you will notice that the actual sleep time is significantly longer.

macOS 10.9 (or higher) systems have a timer coalescing feature, enabled by default. This is a power-saving technique that reduces the precision of software timers, achieving a reduction in CPU usage.

What effect does this have on your *Connext DDS* application? Suppose you send samples from your publisher at a 5 ms rate, using **NDDSUtility_sleep()** to calculate that wait time. You have a subscriber with a deadline set to 6 ms. The timer coalescing feature could make your sleep last much longer than 5-6 ms, so when the next sample reaches the subscriber, the deadline period has expired and you will experience missed samples.

If you are having similar issues, see if your kernel has timer coalescing enabled. You can tell by using this command:

```
user@osx:~$ /usr/sbin/sysctl -a | grep coalescing_enabled
```

In the reply, a 1 means enabled, 0 means disabled.

```
kern.timer.coalescing_enabled: 1
```

To overcome this situation, you must disable timer coalescing in the kernel configuration. (Note that you must have **sudo** or **root** access to be able to edit this kernel parameter.)

```
user@osx:~$ sudo /usr/sbin/sysctl -w kern.timer.coalescing_enabled=0
```

The reply should be:

```
kern.timer.coalescing_enabled: 1 -> 0
```

This change won't be permanent though, and will go back to the default when the system is rebooted.

To make this change permanent, add the configuration line in the file **/etc/sysctl.conf**. You can use your favorite editor to do it, or use this command:

```
user@osx:~$ sudo echo "kern.timer.coalescing_enabled=0" >> /etc/sysctl.conf
```

9.9 Libraries Required for Using Distributed Logger

RTI Distributed Logger is supported on macOS platforms. 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++ (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

9.10 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 in Table 9.8 Additional Libraries for Using Monitoringmust 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.dylib	librtimonitoringd.dylib

9.11 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 macOS architectures. See the RTI Secure WAN Transport Release Notes and RTI Secure WAN Transport Installation Guide.

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 macOS Systems. (Select the files appropriate for your chosen library format.)

Table 9.9 Additional Libraries for using RTI Secure WAN Transport APIs on macOS 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	

9.12 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 macOS Systems . If you are using *RTI TLS Support*, see Table 9.11 Additional Libraries for using RTI TCP Transport APIs on macOS Systems with TLS Enabled. (Select the files appropriate for your chosen library format.)

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

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

Table 9.10 Additional Libraries for using RTI TCP Transport APIs on macOS 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 9.11 Additional Libraries for using RTI TCP Transport APIs on macOS 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

9.13 Libraries Required for Zero Copy Transfer Over Shared Memory

To use the Zero Copy transfer over shared memory feature, link against the additional library in Table 9.12 Additional Libraries for Zero Copy Transfer Over Shared Memory.

Table 9.12 Additional Libraries for Zero Copy Transfer Over Shared Memory

Static Release	Static Debug	Dynamic Release	Dynamic Debug
libnddsmetpz.a	libnddsmetpzd.a	libnddsmetp.so	libnddsmetpd.so

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

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

Chapter 10 QNX Platforms

Table 10.1 Supported QNX Platforms lists the architectures supported on QNX operating systems.[a]

Table 10.1 Supported QNX Platforms

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
	PowerPC™ E500 V2 core *Custom-supported target platform. Contact your RTI sales representative or sales@rti.com for more information.*	qcc (gcc 4.4.2)	ppce500v2QNX6.5.0qcc_cpp4.4.2
QNX Neutrino 6.5 SP1 [b]	Arm v7	qcc 4.4.2 with Dinkum libraries	armv7aQNX6.5.0SP1qcc_cpp4.4.2

[a]For use with Windows, Linux or Solaris host as supported by QNX and RTI.

[b]These libraries require a hardware FPU in the processor and are compatible with systems that have hard-float libc. See platform notes for compiler flags details.

Table 10.1 Supported QNX Platforms

Operating System	CPU	Compiler	RTI Architecture
QNX Neutrino 6.6.0	Arm v7 See note regarding patches below. *Custom-supported target platform. Contact your RTI sales representative or sales@rti.com for more information.*	qcc_cpp 4.7.3 with Dinkumware libraries	armv7aQNX6.6.0qcc_cpp4.7.3
	x86 See note regarding patches below. *Custom-supported target platform. Contact your RTI sales representative or sales@rti.com for more information.*	qcc_cpp 4.7.3 with Dinkumware libraries	i86QNX6.6qcc_cpp4.7.3
QNX Neutrino 7.0 [a]	Arm v7 See note regarding patches below. *Custom-supported target platform. Contact your RTI sales representative or sales@rti.com for more information.*	qcc 7.0.0 with LLVM default libraries	armv7QNX7.0.0qcc_cxx5.4.0
	Arm v8 64-bit	qcc 7.0.0 with LLVM default libraries	armv8QNX7.0.0qcc_cxx5.4.0
		qcc 7.0.0 with GNU C++ libraries	armv8QNX7.0.0qcc_gpp5.4.0
	x64	qcc 7.0.0 with LLVM default libraries	x64QNX7.0.0qcc_cxx5.4.0
		qcc 7.0.0 with GNU C++ libraries	x64QNX7.0.0qcc_gpp5.4.0

Note regarding patches: The libraries for QNX Neutrino 6.6.0 on Arm v7 (armv7aQNX6.6.0qcc_cpp4.7.3) and x86 (i86QNX6.6qcc_cpp4.7.3), and QNX Neutrino 7.0 on Arm v7 (armv7QNX7.0.0qcc_cxx5.4.0) were built with the following patches and tested on systems that have these patches installed:

- QNX Software Development Platform 6.6 Graphics Patch (Patch ID 3875)
- QNX Software Development Platform 6.6 Header Files Patch (Patch ID 3851)
- Patch-660-3885-diskimage.tar for the BSP

[a]These libraries require a hardware FPU in the processor and are compatible with systems that have hard-float libc. See platform notes for compiler flags details.

Table 10.2 Building Instructions for QNX Architectures lists the libraries you will need to link into your application.

See also:

- 10.8 Libraries Required for Using Distributed Logger on page 103
- 10.9 Libraries Required for Using Monitoring on page 104
- 10.10 Libraries Required for Using RTI Secure WAN Transport APIs on page 104
- 10.11 Libraries Required for Using RTI TCP Transport APIs on page 105
- 10.12 Libraries Required for Zero Copy Transfer Over Shared Memory on page 106

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

Table 10.4 Library-Creation Details for QNX Architectures provides details on how the QNX libraries were built.

Starting with release 6.0.1, you will need the **dirname** tool to run the scripts in the **bin** directory.

Table 10.2 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 or librticonnextmsgcpp2z.a	-lm -lsocket	-DRTI_QNX
	Static Debug	libnddscppzd.a or libnddscpp2zd.a libnddsczd.a libnddscorezd.a librticonnextmsgcppzd.a or librticonnextmsgcpp2zd.a		
	Dynamic Release	libnddscpp.so or libnddscpp2.so libnddsc.so libnddscore.so librticonnextmsgcpp.so or librticonnextmsgcpp2.so		
	Dynamic Debug	libnddscppd.so or libnddscpp2d.so libnddscd.so libnddscored.so librticonnextmsgcppd.so or librticonnextmsgcpp2d.so		

[a]Choose *cpp*.* for the Traditional C++ API or *cpp2*.* 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, Secure, Basic, or Evaluation package type. It is not provided with the *RTI Connext DDS* Core package type.

Table 10.2 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 10.3 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 *cpp*.* for the Traditional C++ API or *cpp2*.* 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, Secure, Basic, or Evaluation 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 10.4 Library-Creation Details for QNX Architectures

RTI Architecture	Library Format (Static & Dynamic)	Compiler Flags Used by RTI
armv7aQNX6.5.0SP1 qcc_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-T=\"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-T=\"armv7aQNX6.5.0SP1qcc_cpp4.4.2\"
armv7aQNX6.6.0 qcc_cpp4.7.3	Release	qcc -Vgcc/4.7.3,gcc_ntoarmv7le_cpp -lang-c -fPIC -fexceptions -DFD_SETSIZE=512 -O -Wall -Wno-unknown-pragmas -DPtrIntType=long -DCSREAL_IS_FLOAT -DCPU=ARMV7 -DTARGET-T=\"armv7aQNX6.6.0qcc_cpp4.7.3\" -DNDEBUG
	Debug	qcc -Vgcc/4.7.3,gcc_ntoarmv7le_cpp -lang-c -fPIC -fexceptions -DFD_SETSIZE=512 -g -Wall -Wno-unknown-pragmas -DPtrIntType=long -DCSREAL_IS_FLOAT -DCPU=ARMV7 -DTARGET-T=\"armv7aQNX6.6.0qcc_cpp4.7.3\"
armv7QNX7.0.0 qcc_cxx5.4.0	Release	qcc -Vgcc/5.4.0,gcc_ntoarmv7le -Y_cxx -O -Wall -Wno-unknown-pragmas -fPIC -fexceptions -DFD_SETSIZE=512 - DPtrIntType=long -DCSREAL_IS_FLOAT -DCPU=ARM7 -DTARGET-T=\"armv7QNX7.0.0qcc_cxx5.4.0\" -DNDEBUG
	Debug	qcc -Vgcc/5.4.0,gcc_ntoarmv7le -Y_cxx -g -Wall -Wno-unknown-pragmas -fPIC -fexceptions -DFD_SETSIZE-E=512 - DPtrIntType=long -DCSREAL_IS_FLOAT -DCPU=ARM7 -DTARGET=\"armv7QNX7.0.0qcc_cxx5.4.0\"
armv8QNX7.0.0 qcc_cxx5.4.0	Release	qcc -Vgcc/5.4.0,gcc_ntoaarch64le -Y_cxx -Wall -Wno-unknown-pragmas -fPIC -fexceptions -DFD_SETSIZE=512 -DPtrIntType=long -DCSREAL_IS_FLOAT -DCPU=ARM64 -DTARGET-T=\"armv8QNX7.0.0qcc_cxx5.4.0\" -DNDEBUG
	Debug	qcc -Vgcc/5.4.0,gcc_ntoaarch64le -Y_cxx -g -Wall -Wno-unknown-pragmas -fPIC -fexceptions -DFD_SETSIZE=512 -DPtrIntType=long -DCSREAL_IS_FLOAT -DCPU=ARM64 -DTARGET-T=\"armv8QNX7.0.0qcc_cxx5.4.0\"
armv8QNX7.0.0 qcc_gpp5.4.0	Release	qcc -Vgcc/5.4.0,gcc_ntoaarch64le -Y_gpp -Wall -Wno-unknown-pragmas -fPIC -fexceptions -DFD_SETSIZE=512 -DPtrIntType=long -DCSREAL_IS_FLOAT -DCPU=ARM64 -DTARGET-T=\"armv8QNX7.0.0qcc_gpp5.4.0\" -DNDEBUG
	Debug	qcc -Vgcc/5.4.0,gcc_ntoaarch64le -Y_gpp -g -Wall -Wno-unknown-pragmas -fPIC -fexceptions -DFD_SETSIZE=512 -DPtrIntType=long -DCSREAL_IS_FLOAT -DCPU=ARM64 -DTARGET-T=\"armv8QNX7.0.0qcc_gpp5.4.0\"
i86QNX6.4.1 qcc_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

Table 10.4 Library-Creation Details for QNX Architectures

RTI Architecture	Library Format (Static & Dynamic)	Compiler Flags Used by RTI
i86QNX6.5 qcc_gpp4.4.2	Release	qcc -Vgcc/4.4.2,gcc_ntox86 -Y_gpp -m32 -march=i386 -mtune=generic -fPIC -fexceptions -DFD_SETSIZE-E=512 -O -Wall -Wno-unknown-pragmas -DRTS_QNX -DPtrIntType=long -DCSREAL_IS_FLOAT -DCPU-U=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-E=512 -g -Wall -Wno-unknown-pragmas -DRTS_QNX-DPtrIntType=long -DCSREAL_IS_FLOAT -DCPU-U=I80586 -DTARGET=\"i86QNX6.5qcc_gpp4.4.2\"
i86QNX6.6 qcc_cpp4.7.3	Release	qcc -Vgcc/4.7.3,gcc_ntox86 -Y_cpp -m32 -march=i386 -mtune=generic -lang-c -fPIC -fexceptions -DFD_ SETSIZE=512 -O -Wall -Wno-unknown-pragmas -DPtrIntType=long -DCSREAL_IS_FLOAT -DCPU=I80586 -DTARGET=\"i86QNX6.6qcc_cpp4.7.3\" -DNDEBUG
	Debug	qcc -Vgcc/4.7.3,gcc_ntox86 -Y_cpp -m32 -march=i386 -mtune=generic -lang-c -fPIC -fexceptions -DFD_ SETSIZE=512 -g -Wall -Wno-unknown-pragmas -DPtrIntType=long -DCSREAL_IS_FLOAT -DCPU=I80586 -DTARGET=\"i86QNX6.6qcc_cpp4.7.3\"
ppce500v2QNX6.5.0 qcc_cpp4.4.2	Release	qcc Vgcc/4.4.2,gcc_ntoppcbespe -Y_cpp -v -mcpu=8540 -me500v2 -mno-isel -mspe -mhard-float -Wl,-relax -fPIC -fexceptions -O -DNDEBUG
	Debug	qcc Vgcc/4.4.2,gcc_ntoppcbespe -Y_cpp -v -mcpu=8540 -me500v2 -mno-isel -mspe -mhard-float -Wl,-relax -fPIC -fexceptions -g
x64QNX7.0.0 qcc_cxx5.4.0	Release	qcc -Vgcc/5.4.0,gcc_ntox86_64 -Y_cxx -O -Wall -Wno-unknown-pragmas -fPIC -fexceptions -DFD_ SETSIZE=512 -DPtrIntType=long -DCSREAL_IS_FLOAT -DCPU=AMD64 -DTARGET=\"x64QNX7.0.0qcc_ cxx5.4.0\" -DNDEBUG
	Debug	qcc -Vgcc/5.4.0,gcc_ntox86_64 -Y_cxx -g -Wall -Wno-unknown-pragmas -fPIC -fexceptions -DFD_ SETSIZE=512 -DPtrIntType=long -DCSREAL_IS_FLOAT -DCPU=AMD64 -DTARGET=\"x64QNX7.0.0qcc_ cxx5.4.0\"
x64QNX7.0.0 qcc_gpp5.4.0	Release	qcc -Vgcc/5.4.0,gcc_ntox86_64 -Y_gpp -O -Wall -Wno-unknown-pragmas -fPIC -fexceptions -DFD_ SETSIZE=512 -DPtrIntType=long -DCSREAL_IS_FLOAT -DCPU=AMD64 -DTARGET=\"x64QNX7.0.0qcc_ gpp5.4.0\" -DNDEBUG
	Debug	qcc -Vgcc/5.4.0,gcc_ntox86_64 -Y_gpp -g -O0 -Wall -Wno-unknown-pragmas -fPIC -fexceptions -DFD_ SETSIZE=512 -DPtrIntType=long -DCSREAL_IS_FLOAT -DCPU=AMD64 -DTARGET=\"x64QNX7.0.0qcc_ gpp5.4.0\"

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

For QNX architectures in *Connext DDS* 5.0 and higher:

The C++ libraries are 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.

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 available for all QNX platforms and has been tested with C++03.

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

10.3 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.

10.4 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 all platforms except QNX Neutrino 6.4.1 (i86QNX6.4.1qcc_gpp), QNX Neutrino 7.0.0 on Arm v7 (armv7QNX7.0.0qcc_cxx5.4.0), and QNX Neutrino 6.5 on PPC (ppce500v2QNX6.5.0qcc_cpp4.4.2).

10.5 Unsupported Features

These features are not supported on QNX platforms:

- Controlling CPU Core Affinity
- Durable Writer History and Durable Reader State
- 'Find Package' CMake script

10.6 Monotonic Clock Support

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

10.7 Thread Configuration

See Table 10.5 Thread Settings for QNX Platforms and Table 10.6 Thread-Priority Definitions for QNX Platforms.

Table 10.5 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	64 * 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	64 * 1024
	cpu_list	CPU core affinity not supported
	cpu_rotation	CPU core affinity not supported

Table 10.5 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 * 64 * 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	12
	stack_size	4 * 64 * 1024
	cpu_list	CPU core affinity not supported
	cpu_rotation	CPU core affinity not supported

Table 10.6 Thread-Priority Definitions for QNX Platforms

Thread-Priority Definition	Operating-System Priority
THREAD_PRIORITY_DEFAULT	10
THREAD_PRIORITY_HIGH	14
THREAD_PRIORITY_ABOVE_NORMAL	12
THREAD_PRIORITY_NORMAL	10
THREAD_PRIORITY_BELOW_NORMAL	8
THREAD_PRIORITY_LOW	6

10.8 Libraries Required for Using Distributed Logger

RTI Distributed Logger is supported on all QNX platforms except armv7aQNX6.5.0SP1qcc_cpp4.4.2.

Table 10.7 Additional Libraries for using RTI Distributed Logger lists the additional libraries you will need in order to use *Distributed Logger*.

Table 10.7 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

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 are plan to use *static* libraries, the RTI library from Table 10.8 Additional Libraries for Using Monitoringmust appear *first* in the list of libraries to be linked.

Table 10.8 Additional Libraries for Using Monitoring

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

10.10 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 10.9 Additional Libraries for using RTI Secure WAN Transport APIs on QNX 6.5 Systems. (Select the files appropriate for your chosen library format.)

---

[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.

Table 10.9 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	

10.11 Libraries Required for Using RTI TCP Transport APIs

To use the TCP Transport APIs, link against the additional libraries in Table 10.10 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 10.11 Additional Libraries for using RTI TCP Transport APIs on QNX Systems with TLS Enabled. (Select the files appropriate for your chosen library format.)

Table 10.10 Additional Libraries for using RTI TCP Transport APIs on QNX 6.5 Systems

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

Table 10.11 Additional Libraries for using RTI TCP Transport APIs on QNX Systems with TLS Enabled

Library Format	RTI TLS Libraries[d]
Dynamic Release	libnddstls.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>.

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

Table 10.11 Additional Libraries for using RTI TCP Transport APIs on QNX Systems with TLS Enabled

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

10.12 Libraries Required for Zero Copy Transfer Over Shared Memory

To use the Zero Copy transfer over shared memory feature, link against the additional library in Table 10.12 Additional Libraries for Zero Copy Transfer Over Shared Memory.

Table 10.12 Additional Libraries for Zero Copy Transfer Over Shared Memory

Static Release	Static Debug	Dynamic Release	Dynamic Debug
libnddsmetpz.a	libnddsmetpzd.a	libnddsmetp.so	libnddsmetpd.so

10.13 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>.

Chapter 11 Solaris Platforms

Note: Solaris platforms are only available by request.

Table 11.1 Supported Solaris Platforms lists the architectures supported on Solaris operating systems.

Table 11.1 Supported Solaris Platforms

Operating System	CPU	Compiler or Software Development Kit	RTI Architecture
Solaris 10	UltraSPARC with native 64-bit support	gcc3.4.2	sparc64Sol2.10gcc3.4.2
		Java Platform, Standard Edition JDK 1.8	

Table 11.2 Building Instructions for Solaris Architectures lists the compiler flags and the libraries you will need to link into your application.

See also:

- 11.7 Libraries Required for Using Monitoring on page 114
- 11.8 Libraries Required for using RTI Secure WAN Transport APIs on page 115
- 11.9 Libraries Required for Zero Copy Transfer Over Shared Memory on page 116

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 11.3 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 11.4 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 11.2 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	-DRTI_UNIX -m64
	Static Debug	libnddsczd.a libnddscorezd.a lib-rticonnextmsgczd.a		
	Dynamic Release	libnddsc.so libnddscore.so librticonnextmsgc.so		
	Dynamic Debug	libnddscd.so libnddscored.so librticonnextmsgcd.so		

[a]Choose *cpp*.* for the Traditional C++ API or *cpp2*.* 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, Secure, Basic, or Evaluation package type. It is not provided with the *RTI Connext DDS* Core package type.

Table 11.2 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 or librticonnextmsgcpp2z.a	-ldl -lnsl -lsocket -lgen -lposix4 -lpthread -lm -lc	-DRTI_UNIX -m64
	Static Debug	libnddscppzd.a or libnddscpp2zd.a libnddsczd.a libnddscorezd.a librticonnextmsgcppzd.a or librticonnextmsgcpp2zd.a		
	Dynamic Release	libnddscpp.so or libnddscpp2.so libnddsc.so libnddscore.so librticonnextmsgcpp.so or librticonnextmsgcpp2.so		
	Dynamic Debug	libnddscppd.so or libnddscpp2d.so libnddscd.so libnddscored.so librticonnextmsgcppd.so or librticonnextmsgcpp2d.so		
Java	Release	nddsjava.jar rticonnextmsg.jar	N/A	None required
	Debug	nddsjavad.jar rticonnextmsgd.jar		

[a]Choose *cpp*.* for the Traditional C++ API or *cpp2*.* 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, Secure, Basic, or Evaluation package type. It is not provided with the *RTI Connext DDS* Core package type.

Table 11.3 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 11.4 Library-Creation Details for Solaris Architectures

	Library Format	Compiler Flags Used by RTI
sparc64Sol2.10gcc3.4.2[c]	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 1.5 -source 1.5
	Dynamic Debug	-target 1.5 -source 1.5 -g

[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.

11.1 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 Solaris platforms and has been tested with C++03.

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

11.2 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.

11.3 Supported Transports

- **Shared memory:** Supported and enabled by default.
- **UDPv4:** Supported and enabled by default.
- **UDPv6:** Supported for all Solaris 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.

11.3.1 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 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*.

11.3.2 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.

11.4 Unsupported Features

These features are not supported on Solaris platforms:

- Controlling CPU Core Affinity
- Distributed Logger
- Durable Writer History and Durable Reader State
- 'Find Package' CMake script
- Internal setting of thread names at the operating-system level

11.5 Monotonic Clock Support

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

11.6 Thread Configuration

See Table 11.5 Thread Settings for Solaris Platforms and Table 11.6 Thread-Priority Definitions for Solaris Platforms.

Table 11.5 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 11.5 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 11.6 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	

11.7 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.

Notes:

- Memory usage is not available in monitoring data.
- If you plan to use *static* libraries, the RTI library in Table 11.7 Additional Libraries for Using Monitoring must appear *first* in the list of libraries to be linked.

Table 11.7 Additional Libraries for Using Monitoring

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

11.8 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 11.8 Additional Libraries for using RTI Secure WAN Transport APIs. (Select the files appropriate for your chosen library format.)

Table 11.8 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.

11.9 Libraries Required for Zero Copy Transfer Over Shared Memory

To use the Zero Copy transfer over shared memory feature, link against the additional library in Table 11.9 Additional Libraries for Zero Copy Transfer Over Shared Memory.

Table 11.9 Additional Libraries for Zero Copy Transfer Over Shared Memory

Static Release	Static Debug	Dynamic Release	Dynamic Debug
libnddsmetpz.a	libnddsmetpzd.a	libnddsmetp.so	libnddsmetpd.so

Chapter 12 VxWorks Platforms

Table 12.1 VxWorks Target Platformslists the architectures supported on VxWorks operating systems. You can build a VxWorks application by cross-compiling from your development host.

Table 12.1 VxWorks Target Platforms

Operating System	CPU	Compiler	RTI Architecture[a]
VxWorks 6.9	x86	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 *Custom-supported target platform. Contact your RTI sales representative or sales@rti.com for more information.*	Arm v7	gcc 4.3.3	For Kernel Modules: armv7aVx6.9gcc4.3.3 For Real-Time Processes: armv7aVx6.9gcc4.3.3_rtp
	PPC (e500v2)	gcc 4.3.3	For Kernel Modules: ppce500v2Vx6.9gcc4.3.3 For Real-Time Processes: ppce500v2Vx6.9gcc4.3.3_rtp

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

Table 12.1 VxWorks Target Platforms

Operating System	CPU	Compiler	RTI Architecture[a]
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
	MIPS *Custom-supported target platform. Contact your RTI sales representative or sales@rti.com for more information.*	gcc 4.3.3	For Kernel Modules: mips32r2sfbeVx6.9gcc4.3.3 For Real-Time Processes: mips32r2sfbeVx6.9gcc4.3.3_rtp
VxWorks 6.9.4.2	PPC (e500v2)	gcc 4.3.3	For Kernel Modules: ppce500v2Vx6.9.4gcc4.3.3 For Real-Time Processes: ppce500v2Vx6.9.4gcc4.3.3_rtp
	Any PowerPC CPU with floating-point hardware that is backwards-compatible with 32-bit PowerPC 604	gcc 4.3.3	For Kernel Modules: ppc604Vx6.9.4gcc4.3.3 For Real Time Processes: ppc604Vx6.9.4gcc4.3.3_rtp
VxWorks 6.9.4.6	PowerPC®	Diab 5.9.1	For Kernel Modules: ppce6500Vx6.9.4.6diab5.9.1 For Real-Time Processes: ppce6500Vx6.9.4.6diab5.9.1 _rtp
VxWorks 6.9.4.11 *Custom-supported target platform. Contact your RTI sales representative or sales@rti.com for more information*	Arm v7a	gcc 4.3.3	For Kernel Modules: armv7aVx6.9.4gcc4.3.3 For Real Time Processes: armv7aVx6.9.4gcc4.3.3_rtp
VxWorks 7.0 SR0510	x64	gcc 4.8.1.6	For Kernel Modules: pentium64Vx7.0gcc4.8.1 For Real Time Processes: pentium64Vx7.0gcc4.8.1_rtp
VxWorks 7.0 SR0540 *Custom-supported target platform. Contact your RTI sales representative or sales@rti.com for more information*	x86	gcc 4.8.1.10	pentiumVx7.0SR0540gcc4.8.1.10
VxWorks 653 2.3 Only available by request.	PPC 8641D	gcc 3.3.2	sbc8641Vx653-2.3gcc3.3.2

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

Table 12.1 VxWorks Target Platforms

Operating System	CPU	Compiler	RTI Architecture[a]
VxWorks 653 2.5.0.2 *Custom-supported target platform. Contact your RTI sales representative or sales@rti.com for more information.*	PowerPC® E500v2 CPU with floating point hardware	gcc 4.3.3	ppce500v2Vx653-2.5gcc4.3.3[b]

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

- Table 12.2 Building Instructions for VxWorks 6.x - 7.x Architectures on the next page
- Table 12.3 Building Instructions for VxWorks 653 Architectures on page 122

For other libraries that you may need, see:

- 12.16 Libraries Required for Using Distributed Logger on page 135
- 12.17 Libraries Required for Using Monitoring on page 135
- 12.18 Libraries Required for Zero-Copy Transfer Over Shared Memory on page 136

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 12.4 Library-Creation Details for All VxWorks Architectures on page 123 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*.

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

[b]The set of libraries for this architecture was tested on a p2020 board with a custom BSP (Board Support Package).

Table 12.2 Building Instructions for VxWorks 6.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 or librticonnextmsgcpp2z.a	INCLUDE_TIMESTAMP INCLUDE_POSIX_CLOCKS For RTI architectures with SMP support also use: INCLUDE_TLS	-DRTI_VXWORKS
	Static Debug	libnddscppzd.a or libnddscpp2zd.a libnddsczd.a libnddscorezd.a librticonnextmsgcppzd.a or librticonnextmsgcpp2zd.a		
	Dynamic Release	libnddscpp.so or libnddscpp2.so librticonnextmsgcpp.so or librticonnextmsgcpp2.so libnddsc.so libnddscore.so libnddscpp.so		
	Dynamic Debug	libnddscppd.so or libnddscpp2d.so librticonnextmsgcppd.so or librticonnextmsgcpp2d.so libnddscd.so libnddscored.so libnddscppd.so		

[a]Choose *cpp*.* for the Traditional C++ API or *cpp2*.* 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, Secure, Basic, or Evaluation package type. It is not provided with the *RTI Connext DDS* Core package type.

Table 12.2 Building Instructions for VxWorks 6.x - 7.x Architectures

API	Library Format	Required RTI Libraries[abc]	Required Kernel Components	Required Compiler Flags
C	Static Release	libnddscz.a libnddscorez.a librticonnextmsgcz.a	INCLUDE_TIMESTAMP INCLUDE_POSIX_CLOCKS For RTI architectures with SMP support, 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 *cpp*.* for the Traditional C++ API or *cpp2*.* 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, Secure, Basic, or Evaluation package type. It is not provided with the *RTI Connext DDS* Core package type.

Table 12.3 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 12.3 Notes for VxWorks 653 Platforms on page 128.	For VxWorks 653 v2.3: -DRTI_VXWORKS -DRTI_VX653 For VxWorks 653 v2.5.0.2: -DRTI_VXWORKS -DRTI_VX653=2502
	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 12.3 Notes for VxWorks 653 Platforms on page 128.	For VxWorks 653 v2.3: -DRTI_VXWORKS -DRTI_VX653 For VxWorks 653 v2.5.0.2: -DRTI_VXWORKS -DRTI_VX653=2502
	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 12.4 Library-Creation Details for All VxWorks Architectures

RTI Architecture	Library Format	Compiler Flags Used by RTI
armv7aVx6.9 gcc4.3.3	Static or Dynamic Release	ccarm -t7 -mfpu=vfp -mfloat-abi=softfp -ansi -fno-zero-initialized-in-bss -fno-builtin -fvolatile -mlong-calls -mapcs-frame -DCPU=ARMARCH7 -DTOOL_FAMILY=gnu -DTOOL=gnu -DARM_USE_VFP -DRTI_VFP_TASK -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	ccarm -t7 -mfpu=vfp -mfloat-abi=softfp -ansi -fno-zero-initialized-in-bss -fno-builtin -fvolatile -mlong-calls -mapcs-frame -DCPU=ARMARCH7 -DTOOL_FAMILY=gnu -DTOOL=gnu -DARM_USE_VFP -DRTI_VFP_TASK -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 -Wp,-MD
armv7aVx6.9 gcc4.3.3_rtp	Static or Dynamic Release	ccarm -t7 -mfpu=vfp -mfloat-abi=softfp -ansi -fno-zero-initialized-in-bss -fno-builtin -fvolatile -mlong-calls -mapcs-frame -DCPU=ARMARCH7 -DTOOL_FAMILY=gnu -DTOOL=gnu -mrtp -DARM_USE_VFP -DRTI_VFP_TASK -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	ccarm -t7 -mfpu=vfp -mfloat-abi=softfp -ansi -fno-zero-initialized-in-bss -fno-builtin -fvolatile -mlong-calls -mapcs-frame -DCPU=ARMARCH7 -DTOOL_FAMILY=gnu -DTOOL=gnu -mrtp -DARM_USE_VFP -DRTI_VFP_TASK -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 -Wp,-MD
armv7aVx6.9.4 gcc4.3.3	Static or Dynamic Release	-t7 -mfpu=vfp -mfloat-abi=softfp -ansi -fno-zero-initialized-in-bss -fno-builtin -fvolatile -mlong-calls -mapcs-frame -DCPU=_VX_ARMARCH7 -DTOOL_FAMILY=gnu -DTOOL=gnu -D_WRS_KERNEL -D_WRS_VX_SMP -D_WRS_CONFIG_SMP -DARMEL -DARM_USE_VFP -DIP_PORT_VXWORKS=69 -Wall -Wno-unknown-pragmas -DCPU=ARMARCH7 -DNDEBUG
	Static or Dynamic Debug	-t7 -mfpu=vfp -mfloat-abi=softfp -ansi -fno-zero-initialized-in-bss -fno-builtin -fvolatile -mlong-calls -mapcs-frame -DCPU=_VX_ARMARCH7 -DTOOL_FAMILY=gnu -DTOOL=gnu -D_WRS_KERNEL -D_WRS_VX_SMP -D_WRS_CONFIG_SMP -DARMEL -DARM_USE_VFP -DIP_PORT_VXWORKS=69 -g -Wall -Wno-unknown-pragmas -DCPU=ARMARCH7
armv7aVx6.9.4 gcc4.3.3_rtp	Static or Dynamic Release	-t7 -mfpu=vfp -mfloat-abi=softfp -mrtp -fno-strict-aliasing -fno-zero-initialized-in-bss -fno-builtin -fvolatile-mlong-calls -mapcs-frame-D_VX_CPU=_VX_ARMARCH7 -D_VX_TOOL_FAMILY=gnu -D_VX_TOOL=gnu -DARMEL -DARM_USE_VFP -mrtp -O2 -fno-strict-aliasing -Wall -Wno-unknown-pragmas -DCPU=ARMARCH7 -DNDEBUG
	Static or Dynamic Debug	-t7 -mfpu=vfp -mfloat-abi=softfp -mrtp -fno-strict-aliasing -fno-zero-initialized-in-bss -fno-builtin -fvolatile-mlong-calls -mapcs-frame-D_VX_CPU=_VX_ARMARCH7 -D_VX_TOOL_FAMILY=gnu -D_VX_TOOL=gnu -DARMEL -DARM_USE_VFP -mrtp -g -Wall -Wno-unknown-pragmas -DCPU=ARMARCH7
mips32r2sfbeVx6.9 gcc4.3.3	Static or Dynamic Release	ccmips -G 0 -mno-branch-likely -mips32r2 -mgp32 -mfp32 -EB -msoft-float -DCPU=MIPSI32R2 -DTOOL_FAMILY=gnu -DTOOL=sfgnu -mlong-calls -D_WRS_KERNEL -D__PROTOTYPE_5_0 -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	ccmips -G 0 -mno-branch-likely -mips32r2 -mgp32 -mfp32 -EB -msoft-float -DCPU=MIPSI32R2 -DTOOL_FAMILY=gnu -DTOOL=sfgnu -mlong-calls -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

Table 12.4 Library-Creation Details for All VxWorks Architectures

RTI Architecture	Library Format	Compiler Flags Used by RTI
mips32r2sfbeVx6.9 gcc4.3.3_rtp	Static or Dynamic Release	ccmips -G 0 -mno-branch-likely -mips32r2 -mgp32 -mfp32 -EB -msoft-float -DRTI_GCC4 -DTOOL=sfgnu -mxgot -mlong-calls -DCPU=MIPSI32R2 -DTOOL_FAMILY=gnu -mrtp -mips32r2 -D__PROTOTYPE_5_0 -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	ccmips -G 0 -mno-branch-likely -mips32r2 -mgp32 -mfp32 -EB -msoft-float -DRTI_GCC4 -DTOOL=sfgnu -mxgot -mlong-calls -DCPU=MIPSI32R2 -DTOOL_FAMILY=gnu -mrtp -mips32r2 -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
pentium64Vx6.9 gcc4.3.3	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
pentium64Vx6.9 gcc4.3.3_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
pentium64Vx7.0 gcc4.8.1	Static or Dynamic Release	-march=atom -mpopcnt -nostdlib -fno-builtin -fno-defer-pop -m64 -fno-omit-frame-pointer -mcmodel=kernel -mno-red-zone -fno-implicit-fp -ansi -fno-zero-initialized-in-bss -Wall -DTOOL_FAMILY=gnu -DTOOL=gnu -D_WRS_KERNEL -O -Wall -DPtrIntType=long -DCSREAL_IS_FLOAT -DCPU=PENTIUM -DNDEBUG
	Static or Dynamic Debug	-march=atom -mpopcnt -nostdlib -fno-builtin -fno-defer-pop -m64 -fno-omit-frame-pointer -mcmodel=kernel -mno-red-zone -fno-implicit-fp -ansi -fno-zero-initialized-in-bss -Wall -DTOOL_FAMILY=gnu -DTOOL=gnu -D_WRS_KERNEL -g -Wall -DPtrIntType=long -DCSREAL_IS_FLOAT -DCPU=PENTIUM
pentium64Vx7.0 gcc4.8.1_rtp	Static or Dynamic Release	-march=atom -mpopcnt -m64 -mcmodel=small -fno-implicit-fp -fno-builtin -fno-omit-frame-pointer -mrtp -fno-strict-aliasing -D_C99 -D_HAS_C9X -fasm -ansi -D_VX_TOOL_FAMILY=gnu -D_VX_TOOL=gnu -O -Wall -DPtrIntType=long -DCSREAL_IS_FLOAT -DCPU=X86_64 -DNDEBUG
	Static or Dynamic Debug	-march=atom -mpopcnt -m64 -mcmodel=small -fno-implicit-fp -fno-builtin -fno-omit-frame-pointer -mrtp -fno-strict-aliasing -D_C99 -D_HAS_C9X -fasm -ansi -D_VX_TOOL_FAMILY=gnu -D_VX_TOOL=gnu -g -Wall -DPtrIntType=long -DCSREAL_IS_FLOAT -DCPU=X86_64

Table 12.4 Library-Creation Details for All VxWorks Architectures

RTI Architecture	Library Format	Compiler Flags Used by RTI
pentiumVx6.9 gcc4.3.3	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
pentiumVx6.9 gcc4.3.3_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
pentiumVx7.0SR0540 gcc4.8.1.10	Static or Dynamic Release	-march=pentium4 -nostdlib -fno-omit-frame-pointer -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 -D_WRS_CONFIG_SMP -DVXWORKS_MAJOR_VERSION=7 -DVXWORKS_MINOR_VERSION=0 -DTARGET=\"pentiumVx7.0SR0540gcc4.8.1.10\" -O -Wall -Wno-unknown-pragmas -DRTI_VXWORKS -DPtrIntType=long -DCSREAL_IS_FLOAT -DCPU=PENTIUM4 -DRTI_ENDIAN_LITTLE -DRTI_MULTICAST -DRTI_SHARED_MEMORY -DRTI_IPV6 -DRTI_SMP_SUPPORT -DNDEBUG -Wp,-MD
	Static or Dynamic Debug	-march=pentium4 -nostdlib -fno-omit-frame-pointer -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 -D_WRS_CONFIG_SMP -g -DRTI_PRECONDITION_TEST -DVXWORKS_MAJOR_VERSION=7 -DVXWORKS_MINOR_VERSION=0 -DTARGET=\"pentiumVx7.0SR0540gcc4.8.1.10\" -Wall -Wno-unknown-pragmas -DRTI_VXWORKS -DPtrIntType=long -DCSREAL_IS_FLOAT -DCPU=PENTIUM4 -DRTI_ENDIAN_LITTLE -DRTI_MULTICAST -DRTI_SHARED_MEMORY -DRTI_IPV6 -DRTI_SMP_SUPPORT -Wp,-MD
ppc604Vx6.9 gcc4.3.3, ppc604Vx6.9.4 gcc4.3.3	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
ppc604Vx6.9 gcc4.3.3_rtp, ppc604Vx6.9.4 gcc4.3.3_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 12.4 Library-Creation Details for All VxWorks Architectures

RTI Architecture	Library Format	Compiler Flags Used by RTI
ppce500v2Vx6.9 gcc4.3.3	Static or Dynamic Release	ccppc -m32 -mstrict-align -ansi -fno-builtin -mlongcall -DCPU=PPC32 -DTOOL_FAMILY=gnu -DTOOL-L=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-L=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 gcc4.3.3_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
ppce500v2Vx6.9.4 gcc4.3.3	Static or Dynamic Release	ccppc -m32 -mstrict-align -ansi -fno-builtin -mlongcall -DCPU=PPC32 -DTOOL_FAMILY=gnu -DTOOL-L=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-L=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
ppce500v2Vx6.9.4 gcc4.3.3_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
ppce500v2Vx653-2.5gcc4.3.3	Static or Dynamic Release	-DVTHREADS -DTOOL_FAMILY=gnu -DTOOL=gnu -mcpu=powerpc -m32 -DCPU=PPC604 -Wall -B$(WIND_HOME)/gnu/4.3.3-vxworks653/lib/gcc -D_WRS_KERNEL -D__PROTOTYPE_5_0 -DVXWORKS_MAJOR_VERSION=5 -DVXWORKS_MINOR_VERSION=5 -O -Wall -Wno-unknown-pragmas -DRTS_VXWORKS -DPtrIntType=long -DCSREAL_IS_FLOAT -Wp,-MD -Wp -NDEBUG
	Static or Dynamic Debug	-DVTHREADS -DTOOL_FAMILY=gnu -DTOOL=gnu -mcpu=powerpc -m32 -DCPU=PPC604 -Wall -B$(WIND_HOME)/gnu/4.3.3-vxworks653/lib/gcc -D_WRS_KERNEL -D__PROTOTYPE_5_0 -DVXWORKS_MAJOR_VERSION=5 -DVXWORKS_MINOR_VERSION=5 -O -Wall -Wno-unknown-pragmas -DRTS_VXWORKS -DPtrIntType=long -DCSREAL_IS_FLOAT -Wp,-MD -Wp

Table 12.4 Library-Creation Details for All VxWorks Architectures

RTI Architecture	Library Format	Compiler Flags Used by RTI
ppce6500Vx6.9.4.6 diab5.9.1	Static or Dynamic Release	-tPPCE6500FV:vxworks69 -DCPU=_VX_PPC85XX -DTOOL_FAMILY=diab -DTOOL=diab -D_WRS_KERNEL -DCPU_VARIANT=_e6500 -Xkeywords=0x0fffffde -ew1547,1551,1552,1554 -I/data/toolchains/vxworks/vx-6.9.4.6/vxworks-6.9/target/h -I/data/toolchains/vxworks/vx-6.9.4.6/vxworks-6.9/target/h/wrn/coreip -Xforce-prototypes -DNDEBUG
	Static or Dynamic Debug	-tPPCE6500FV:vxworks69 -DCPU=_VX_PPC85XX -DTOOL_FAMILY=diab -DTOOL=diab -D_WRS_KERNEL -DCPU_VARIANT=_e6500 -Xkeywords=0x0fffffde -ew1547,1551,1552,1554 -I/data/toolchains/vxworks/vx-6.9.4.6/vxworks-6.9/target/h -I/data/toolchains/vxworks/vx-6.9.4.6/vxworks-6.9/target/h/wrn/coreip -g -Xdebug-struct-all -Xforce-prototypes
ppce6500Vx6.9.4.6 diab5.9.1_rtp	Static or Dynamic Release	-tPPCEH:rtp -DCPU=_VX_PPC32 -D_VX_TOOL_FAMILY=diab -D_VX_TOOL=diab -WDVSB_DIR-R=/data/toolchains/vxworks/vx-6.9.4.6/vxworks-6.9/target/lib -Xansi -Xstsw-slow -Xstmw-slow -Xforce-declarations -Xmake-dependency=0xd -I/data/toolchains/vxworks/vx-6.9.4.6/vxworks-6.9/target/usr/h -I/data/toolchains/vxworks/vx-6.9.4.6/vxworks-6.9/target/h/wrn/coreip -Xforce-prototypes -DNDEBUG
	Static or Dynamic Debug	-tPPCEH:rtp -DCPU=_VX_PPC32 -D_VX_TOOL_FAMILY=diab -D_VX_TOOL=diab -WDVSB_DIR-R=/data/toolchains/vxworks/vx-6.9.4.6/vxworks-6.9/target/lib -Xansi -Xstsw-slow -Xstmw-slow -Xforce-declarations -Xmake-dependency=0xd -I/data/toolchains/vxworks/vx-6.9.4.6/vxworks-6.9/target/usr/h -I/data/toolchains/vxworks/vx-6.9.4.6/vxworks-6.9/target/h/wrn/coreip -g -Xdebug-struct-all -Xforce-prototypes
sbc8641Vx653-2.3 gcc3.3.2	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

12.1 Notes for VxWorks 6 and 7 Platforms

- Known Defect
  - A too-small SO_RCVBUF causes packet loss in polled receiving in VxWorks SMP

    Due to a potential bug in the VxWorks Network stack (V7NET-2540), creating a socket with a very small SO_RCVBUF might cause packet loss when receiving over that socket in a non-blocking way (polling). This problem has been reproduced in SMP kernels.

    To work around this, create the receiving sockets with a SO_RCVBUF size of at least 4000. This is twice the minimum size allowed by VxWorks (IPNET_MIN_RCVBUF_SIZE, which is 2000).

12.2 Notes for VxWorks 7 Platforms

- Required Makefile Change

  For VxWorks 7.0 platforms only: After you run *rtiddsgen*, either edit the generated makefile to specify which VxWorks Source Build (VSB) you want to use or set an environment variable called VSB_DIR that points to the VSB. In the generated makefile, find this line and change it to match your VSB directory:

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

  Note: RTI uses a VSB based on the itl_generic BSP (provided by Wind River) to build the *Connext DDS* libraries for VxWorks 7.0 x64.

- Known Defects
  - When using VxWorks 7.0 64-bit RTP mode, there is a bug in the **getsockopt()** function: the **optlen** parameter is not properly set. Refer to Wind River defect V7NET-1293
  - When using VxWorks 7.0 64-bit RTP mode, an incorrect number of sections is introduced in the resulting ELF binaries, so the VxWorks kernel cannot load them. Refer to Wind River defect VXW7-3771.

12.3 Notes for VxWorks 653 Platforms

- For VxWorks 653 v2.3 platforms: See Table 12.11 Required Kernel Components for VxWorks 653 v 2.3 on page 137.
- For VxWorks 653 v2.5.0.2 platforms: Add the following on top of the default components added by Wind River to support your architecture:
  - INCLUDE_DEBUG_UTIL
  - INCLUDE_FACE_POSIX_SOCKET_DRV
- VxWorks 653 platforms do not support the Zero Copy transfer over shared memory feature.
- See also: 12.15 Memory Settings for VxWorks 653 2.x Platforms on page 134 and 12.11 Unsupported Features on page 132.

12.4 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 Community portal, accessible from https://community.rti.com/kb.

12.5 Dynamic Libraries for Kernel Mode on VxWorks Systems

The *Connext DDS* Professional, Research, Evaluation, and Basic packages include support for the Request-Reply Communication Pattern, for all platforms in Table 12.1 VxWorks Target Platforms on page 117 and all programming languages, with one exception: VxWorks 6.9.4.6 supports Request-Reply in the C API only.

When using a *Connext DDS* dynamic library for 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 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 proper 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 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 request/reply, in addition to the standard *Connext DDS* libraries for core, C, and C++.

12.6 Libraries for RTP Mode on VxWorks Systems

Dynamic libraries are *not* available for VxWorks 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 systems with RTP mode on Pentium CPUs.

12.7 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.

12.8 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 653 and VxWorks 6.9.4.6.

The supported platforms have been tested with both C++03 and C++11. C++ 03 is typically supported with gcc 3.4.2 and above. C++11 is typically supported with gcc 4.7.2.

Both the default and STL plugins are supported, with this exception:

- For VxWorks 6.9.4 on PPC e500v2 (ppce500v2Vx6.9.4gcc4.3.3), only the default plugin is supported for C++03 or C++11.

For VxWorks 7.0 on x64 (pentium64Vx7.0gcc4.8.1), C++03 is supported in kernel mode and C++11 is supported in RTP mode.

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

12.9 Multicast Support

Multicast is supported by VxWorks 6.9.x and 7.x. It is also supported by VxWorks 653 2.3.x (as long as you use a third-party socket library, for details, please contact Wind River Services or RTI Support) and VxWorks 653 2.5.x (this version includes a socket library by default).

Multicast 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.

Known Defects:

- If you have a Wind River account, you can find more information about defect VXW6-19089 (also known as WIND00418701) here: https://support2.windriver.com/index.php?page=defects&on=view&id=VXW6-19089.

  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 VXW6-80771 here: https://support2.windriver.com/index.php?page=defects&on=view&id=VXW6-80771.

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.

12.10 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 653 platforms. See also:
  - 12.10.1 Shared-Memory Communication between Applications Running in Kernel Mode and RTP Requires Explicitly Set Participant ID below
  - 12.10.2 How To Run Connext DDS Libraries in Kernels Built without Shared Memory below
- **UDPv4:** Supported and enabled by default.
- **UDPv6:** Supported on VxWorks 6.7 and higher platforms, except VxWorks 6.9.3, 6.9.3.2 on MIPS CPU, 6.9.4.6, and VxWorks 653. No Traffic Class support.
- **TCP/IPv4:** Not supported.

12.10.1 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.

12.10.2 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.

12.11 Unsupported Features

The VxWorks 653 2.3 (sbc8641Vx653-2.3gcc3.3.2) platform does not support the Zero-Copy transfer over shared memory feature or the RTI FlatData language binding.

The Modern C++ API for Request-Reply Communication is not supported on VxWorks 653 and VxWorks 6.9.4.6 platforms. The Traditional C++ API for Request-Reply is supported.

These features are not supported on any VxWorks platforms:

- Controlling CPU Core Affinity
- Durable Writer History and Durable Reader State
- 'Find Package' CMake script

12.12 Monotonic Clock Support

The monotonic clock (described in "Clock Selection" in the User's Manual) is supported on VxWorks 6.x and higher platforms. This feature is not supported on VxWorks 653 platforms.

12.13 Use of Real-Time Clock

Starting with 5.3.0, *Connext DDS* uses the Real Time Clock to get the time from the System Clock on VxWorks 6.x and higher platforms. Previously **tickGet()** was used for the system clock.

12.14 Thread Configuration

See these tables:

- Table 12.5 Thread Setting for VxWorks Platforms (Applies to Kernel Tasks or Real-Time Process Threads) below
- Table 12.6 Thread-Priority Definitions for VxWorks Platforms on the next page
- Table 12.7 Thread Kinds for VxWorks Platforms on the next page

Table 12.5 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
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 12.6 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 12.7 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

12.15 Memory Settings for VxWorks 653 2.x Platforms

Connext DDS 6.0.1 has been tested with the following memory settings to run a simple C++ HelloWorld application using *Connext DDS* libraries for VxWorks 653 2.x. Please use these settings as a guideline only, as your requirements may vary depending on your use case and platform:

For the Application:

```
<ApplicationDescription>
<MemorySize
 MemorySizeBss="0x5000"
 MemorySizeText="0xFb0000"
 MemorySizeData="f000"
 MemorySizeRoData="0x220000"/>
```

[a]See VxWorks manuals for additional information.

For the POS:

```
<SharedLibraryDescription>
<MemorySize
 MemorySizeBss="0x60000"
 MemorySizeText="0x7000"
 MemorySizeData="8000"
 MemorySizeRoData="0x6000"/>"
```

12.16 Libraries Required for Using Distributed Logger

RTI Distributed Logger is supported all VxWorks architectures except:

- VxWorks 6.9 (pentiumVx6.9gcc4.3.3[_rtp], ppc604Vx6.9gcc4.3.3[_rtp])
- VxWorks 653 (sbc8641Vx653-2.3gcc3.3.2)

Table 12.8 Additional Libraries for using RTI Distributed Logger lists the additional libraries you will need in order to use *Distributed Logger*.

Table 12.8 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

12.17 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.

Notes:

- Automatic loading of the dynamic monitoring library through QoS is not supported.
- Memory and CPU usage is not available in monitoring data.
- If you plan to use *static* libraries, the RTI library from Table 12.9 Additional Libraries for Using Monitoring must appear *first* in the list of libraries to be linked.

Table 12.9 Additional Libraries for Using Monitoring

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

12.18 Libraries Required for Zero-Copy Transfer Over Shared Memory

To use the Zero-Copy Transfer Over Shared Memory feature, link against the additional library in Table 12.10 Additional Libraries for Zero-Copy Transfer Over Shared Memory.

Note: This feature is not supported on VxWorks 653 platforms.

Table 12.10 Additional Libraries for Zero-Copy Transfer Over Shared Memory

Static Release	Static Debug	Dynamic Release	Dynamic Debug
libnddsmetpz.a	libnddsmetpzd.a	libnddsmetp.so	libnddsmetpd.so

12.19 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.

12.20 Required Kernel Components for VxWorks 653 v 2.3

Table 12.11 Required Kernel Components for VxWorks 653 v 2.3[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
INCLUDE_KERNEL_BASIC_INIT	INCLUDE_NETWRS_INETLIB
INCLUDE_KERNEL_BASIC_INIT2	INCLUDE_NETWRS_NETBUFLIB
INCLUDE_KERNEL_CORE	INCLUDE_NETWRS_REMLIB

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

Table 12.11 Required Kernel Components for VxWorks 653 v 2.3[b]

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
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]

[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 13 Windows Platforms

First, see the basic instructions for compiling on Windows systems in the RTI Connext DDS Core Libraries User's Manual (see the chapter on Building Applications).

The following tables provide supplemental information. Table 13.1 Supported Windows Platforms lists the architectures supported on Windows operating systems.

Table 13.1 Supported Windows Platforms

Operating System	CPU	Visual Studio® Version	RTI Architecture Abbreviation	.NET Version[a]	JDK Version
Windows 8	x86	VS 2012 Update 4	i86Win32VS2012	4.5	1.8
		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	1.8
Windows 10	x86	VS 2015 Update 3	i86Win32VS2015	4.6	1.8
		VS 2017 Update 2[b] VS 2019 Version 16.0.0	i86Win32VS2017	4.6.1	

[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]According to Microsoft, anything that works on this platform will also work on Windows IoT Enterprise with the Windows native application.

Table 13.1 Supported Windows Platforms

Operating System	CPU	Visual Studio® Version	RTI Architecture Abbreviation	.NET Version[a]	JDK Version
	x64	VS 2015 Update 3	x64Win64VS2015	4.6	
		VS 2017 Update 2 VS 2019 Version 16.0.0	x64Win64VS2017	4.6.1	
Windows Server 2012 R2	x64	VS 2012 Update 4	x64Win64VS2012	4.5	1.8
		VS 2013 Update 4	x64Win64VS2013	4.5.1	
		VS 2015 Update 3	x64Win64VS2015	4.6	
Windows Server 2016	x86	VS 2015 Update 3	i86Win32VS2015	4.6	1.8
		VS 2017 Update 2 VS 2019 Version 16.0.0	i86Win32VS2017	4.6.1	
	x64	VS 2015 Update 3	x64Win64VS2015	4.6	
		VS 2017 Update 2 VS 2019 Version 16.0.0	x64Win64VS2017	4.6.1	

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

- Table 13.2 Building Instructions for Windows Host Architectures
- Table 13.3 Building Instructions for Windows Target Architectures

See also:

- 13.13 Libraries Required for Using Distributed Logger Support on page 157
- 13.14 Libraries Required for Using Monitoring on page 157
- 13.15 Libraries Required for Using RTI Secure WAN Transport APIs on page 158
- 13.16 Libraries Required for Using RTI TCP Transport APIs on page 158
- 13.17 Libraries Required for Zero Copy Transfer Over Shared Memory on page 159

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

[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#.

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.

Table 13.4 Running Instructions for Windows Architectures on page 146 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 13.5 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 13.2 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" /MD
	Static Debug	nddsczd.lib nddscorezd.lib rticonnextmsgczd.lib		/D "RTI_WIN32" /MDd
	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 ***cpp*.*** for the Traditional C++ API or ***cpp2*.*** 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] For C++/CLI and C#: The **nddsdotnet** and **rticonnextmsgdotnet** library names include a <*version*>, which depends on your version of .NET. Use the digits, such as 451 or 46. See Table 13.1 Supported Windows Platforms on page 139 for supported .NET versions.

Table 13.2 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 or rticonnextmsgcpp2z.lib	netapi32.lib advapi32.lib user32.lib ws2_32.lib	/D "RTI_WIN32" /MD
	Static Debug	nddscppzd.lib or nddscpp2zd.lib nddsczd.lib nddscorezd.lib rticonnextmsgcppzd.lib or rticonnextmsgcpp2zd.lib		/D "RTI_WIN32" /MDd
	Dynamic Release	nddscpp.lib or nddscpp2.lib nddsc.lib nddscore.lib rticonnextmsgcpp.lib or rticonnextmsgcpp2.lib		/D "RTI_WIN32" /D "NDDS_DLL_ VARIABLE" /MD
	Dynamic Debug	nddscppd.lib or nddscpp2d.lib nddscd.lib nddscored.lib rticonnextmsgcppd.lib or rticonnextmsgcpp2d.lib		/D "RTI_WIN32" /D "NDDS_DLL_ VARIABLE" /MDd

[a] Choose ***cpp*.*** for the Traditional C++ API or ***cpp2*.*** 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] For C++/CLI and C#: The **nddsdotnet** and **rticonnextmsgdotnet** library names include a *<version>*, which depends on your version of .NET. Use the digits, such as 451 or 46. See Table 13.1 Supported Windows Platforms on page 139 for supported .NET versions.

Table 13.2 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 ***cpp*.*** for the Traditional C++ API or ***cpp2*.*** 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] For C++/CLI and C#: The **nddsdotnet** and **rticonnextmsgdotnet** library names include a <*version*>, which depends on your version of .NET. Use the digits, such as 451 or 46. See Table 13.1 Supported Windows Platforms on page 139 for supported .NET versions.

Table 13.3 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 /MD /D "WIN32" /D "RTI_WIN32" /D "NDEBUG"
	Static Debug	nddsczd.lib nddscorezd.lib rticonnextmsgczd.lib		/Gd /MDd /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, Secure, Basic, or Evaluation package type. It is not provided with the *RTI Connext DDS* Core package type.

[c]For C++/CLI and C#: The **nddsdotnet** and **rticonnextmsgdotnet** library names include a *<version>*, which depends on your version of .NET. Use the digits, such as 451 or 46. See Table 13.1 Supported Windows Platforms on page 139 for supported .NET versions.

Table 13.3 Building Instructions for Windows Target Architectures

API	Library Format	RTI Libraries or Jar Files [a] [b] [c]	Required System Libraries	Required Compiler Flags
C++ (Traditional and Modern APIs)	Static Release	nddscppz.lib or nddscpp2z.lib nddscz.lib nddscorez.lib rticonnextmsgcppz.lib or rticonnextmsgcpp2z.lib	netapi32.lib advapi32.lib user32.lib ws2_32.lib	/Gd /EHsc /MD /D "WIN32" /D "RTI_WIN32" /D "NDEBUG"
	Static Debug	nddscppzd.lib or nddscpp2zd.lib nddsczd.lib nddscorezd.lib rticonnextmsgcppzd.lib or rticonnextmsgcpp2zd.lib		/Gd /EHsc /MDd /D "WIN32" /D "RTI_WIN32"
	Dynamic Release	nddscpp.lib or nddscpp2.lib nddsc.lib nddscore.lib rticonnextmsgcpp.lib or rticonnextmsgcpp2.lib		/Gd /EHsc /MD /D "WIN32" /D "NDDS_DLL_ VARIABLE" /D "RTI_WIN32" /D "NDEBUG"
	Dynamic Debug	nddscppd.lib or nddscpp2d.lib nddscd.lib nddscored.lib rticonnextmsgcppd.lib or rticonnextmsgcpp2d.lib		/Gd /EHsc /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, Secure, Basic, or Evaluation package type. It is not provided with the *RTI Connext DDS* Core package type.

[c]For C++/CLI and C#: The **nddsdotnet** and **rticonnextmsgdotnet** library names include a <*version*>, which depends on your version of .NET. Use the digits, such as 451 or 46. See Table 13.1 Supported Windows Platforms on page 139 for supported .NET versions.

Table 13.3 Building Instructions for Windows Target Architectures

API	Library Format	RTI Libraries or Jar Files [a] [b] [c]	Required System Libraries	Required Compiler Flags
C#	Release	nddsdotnet<*version*>.dll rticonnextmsgdotnet<*version*>.dll	N/A	N/A
	Debug	nddsdotnet<*version*>d.dll rticonnextmsgdotnet<*version*>d.dll		
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 13.4 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%

[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, Secure, Basic, or Evaluation package type. It is not provided with the *RTI Connext DDS* Core package type.

[c] For C++/CLI and C#: The **nddsdotnet** and **rticonnextmsgdotnet** library names include a <*version*>, which depends on your version of .NET. Use the digits, such as 451 or 46. See Table 13.1 Supported Windows Platforms on page 139 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 13.4 Running Instructions for Windows Architectures

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

Table 13.5 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 1.5 -source 1.5
	Dynamic Debug	-target 1.5 -source 1.5 -g
All 64-bit Windows architectures for Java	Dynamic Release	-target 1.5 -source 1.6
	Dynamic Debug	-target 1.5 -source 1.6 -g

[a] **%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 13.5 Library-Creation Details for Windows Architectures

RTI Architecture	Library Format	Compiler Flags Used by RTI
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 /MD /nodefaultlib:"libcmt.lib" /defaultlib:"msvcrt.lib" /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 /MDd /nodefaultlib:"libcmtd.lib" /defaultlib:"msvcrtd.lib" /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
i86Win32VS2013	Static Release	/W3 -DSTDC99 /FS -DPtrIntType=long -DCSREAL_IS_FLOAT -DCPU=I80586 -DTARGET=\"i86Win32VS2013\" -DWIN32 -D_WINDOWS -D_WIN32_WINNT=0x0501 -DWIN32_LEAN_AND_MEAN /O2 /Zi /MD /nodefaultlib:"libcmt.lib" /defaultlib:"msvcrt.lib" /EHsc -D_CRT_SECURE_NO_DEPRECATE -DNDEBUG -c
	Dynamic Release	/W3 -DSTDC99 /FS -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	/W3 -DSTDC99 /FS -DPtrIntType=long -DCSREAL_IS_FLOAT -DCPU=I80586 -DTARGET=\"i86Win32VS2013\" -DWIN32 -D_WINDOWS -D_WIN32_WINNT=0x0501 -DWIN32_LEAN_AND_MEAN /Od /ZI /MDd /nodefaultlib:"libcmtd.lib" /defaultlib:"msvcrtd.lib" /EHsc /RTC1 -D_CRT_SECURE_NO_DEPRECATE -c
	Dynamic Debug	/W3 -DSTDC99 /FS -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	/W3 -DSTDC99 /FS -DPtrIntType=long -DCSREAL_IS_FLOAT -DCPU=I80586 -DTARGET=\"x86Win32VS2015\" -DWIN32 -D_WINDOWS -D_WIN32_WINNT=0x0501 -DWIN32_LEAN_AND_MEAN /O2 /Zi /MD /nodefaultlib:"libcmt.lib" /defaultlib:"msvcrt.lib" /EHsc -D_CRT_SECURE_NO_DEPRECATE -DNDEBUG -c
	Dynamic Release	/W3 -DSTDC99 -DPtrIntType=long -DCSREAL_IS_FLOAT -DCPU=I80586 -DTARGET=\"x86Win32VS2015\" -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 -DSTDC99 -DPtrIntType=long -DCSREAL_IS_FLOAT -DCPU=I80586 -DTARGET=\"x86Win32VS2015\" -DWIN32 -D_WINDOWS -D_WIN32_WINNT=0x0501 -DWIN32_LEAN_AND_MEAN /Od /ZI /MDd /nodefaultlib:"libcmtd.lib" /defaultlib:"msvcrtd.lib" /EHsc /RTC1 -D_CRT_SECURE_NO_DEPRECATE -c
	Dynamic Debug	/W3 -DSTDC99 -DPtrIntType=long -DCSREAL_IS_FLOAT -DCPU=I80586 -DTARGET=\"x86Win32VS2015\" -DWIN32 -D_WINDOWS -D_WIN32_WINNT=0x0501 -DWIN32_LEAN_AND_MEAN /Od /ZI /MDd /EHsc /RTC1 -D_CRT_SECURE_NO_DEPRECATE -c

Table 13.5 Library-Creation Details for Windows Architectures

RTI Architecture	Library Format	Compiler Flags Used by RTI
i86Win32VS2017	Static Release	/W3 -DSTDC99 /FS -DPtrIntType=long -DCSREAL_IS_FLOAT -DCPU=I80586 -DTARGET=\"x86Win32VS2017\" -DWIN32 -D_WINDOWS -D_WIN32_WINNT=0x0501 -DWIN32_LEAN_AND_MEAN /O2 /Zi /MD /nodefaultlib:"libcmt.lib" /defaultlib:"msvcrt.lib" /EHsc -D_CRT_SECURE_NO_DEPRECATE -DNDEBUG -c
	Dynamic Release	/W3 -DSTDC99 /FS -DPtrIntType=long -DCSREAL_IS_FLOAT -DCPU=I80586 -DTARGET=\"x86Win32VS2017\" -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 -DSTDC99 /FS -DPtrIntType=long -DCSREAL_IS_FLOAT -DCPU=I80586 -DTARGET=\"x86Win32VS2017\" -DWIN32 -D_WINDOWS -D_WIN32_WINNT=0x0501 -DWIN32_LEAN_AND_MEAN /Od /Zl /MDd /nodefaultlib:"libcmtd.lib" /defaultlib:"msvcrtd.lib" /EHsc /RTC1 -D_CRT_SECURE_NO_DEPRECATE -c
	Dynamic Debug	/W3 -DSTDC99 /FS -DPtrIntType=long -DCSREAL_IS_FLOAT -DCPU=I80586 -DTARGET=\"x86Win32VS2017\" -DWIN32 -D_WINDOWS -D_WIN32_WINNT=0x0501 -DWIN32_LEAN_AND_MEAN /Od /Zl /MDd /EHsc /RTC1 -D_CRT_SECURE_NO_DEPRECATE -c
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 /MD /nodefaultlib:"libcmt.lib" /defaultlib:"msvcrt.lib" /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 /Zl /MDd /nodefaultlib:"libcmtd.lib" /defaultlib:"msvcrtd.lib" /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 /Zl /MDd /EHsc /RTC1 -D_CRT_SECURE_NO_DEPRECATE -c

Table 13.5 Library-Creation Details for Windows Architectures

RTI Architecture	Library Format	Compiler Flags Used by RTI
x64Win64VS2013 Note: linker requires /MACHINE:X64 option.	Static Release	/W3 -DSTDC99 /FS -DPtrIntType=long -DCSREAL_IS_FLOAT -DCPU=AMD64 -DTARGET=\"x64Win64VS2013\" -DWIN32 -D_WINDOWS -D_WIN32_WINNT=0x0501 -DWIN32_LEAN_AND_MEAN /O2 /Zi /MD /nodefaultlib:"libcmt.lib" /defaultlib:"msvcrt.lib" /EHsc -D_CRT_SECURE_NO_DEPRECATE -DNDEBUG -c
	Dynamic Release	/W3 -DSTDC99 /FS -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 -DSTDC99 /FS -DPtrIntType=long -DCSREAL_IS_FLOAT -DCPU=AMD64 -DTARGET=\"x64Win64VS2013\" -DWIN32 -D_WINDOWS -D_WIN32_WINNT=0x0501 -DWIN32_LEAN_AND_MEAN /Od /ZI /MDd /nodefaultlib:"libcmtd.lib" /defaultlib:"msvcrtd.lib" /EHsc /RTC1 -D_CRT_SECURE_NO_DEPRECATE -c
	Dynamic Debug	/W3 -DSTDC99 /FS -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
x64Win64VS2015 Note: linker requires /MACHINE:X64 option.	Static Release	/W3 -DSTDC99 /FS -DPtrIntType=long -DCSREAL_IS_FLOAT -DCPU=AMD64 -DTARGET=\"x64Win64VS2015\" -DWIN32 -D_WINDOWS -D_WIN32_WINNT=0x0501 -DWIN32_LEAN_AND_MEAN /O2 /Zi /MD /nodefaultlib:"libcmt.lib" /defaultlib:"msvcrt.lib" /EHsc -D_CRT_SECURE_NO_DEPRECATE -DNDEBUG -c
	Dynamic Release	/W3 -DSTDC99 /FS -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 -DSTDC99 /FS -DPtrIntType=long -DCSREAL_IS_FLOAT -DCPU=AMD64 -DTARGET=\"x64Win64VS2015\" -DWIN32 -D_WINDOWS -D_WIN32_WINNT=0x0501 -DWIN32_LEAN_AND_MEAN /Od /ZI /MDd /nodefaultlib:"libcmtd.lib" /defaultlib:"msvcrtd.lib" /EHsc /RTC1 -D_CRT_SECURE_NO_DEPRECATE -c
	Dynamic Debug	/W3 -DSTDC99 /FS -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
x64Win64VS2017 Note: linker requires /MACHINE:X64 option.	Static Release	/W3 -DSTDC99 /FS -DPtrIntType=long -DCSREAL_IS_FLOAT -DCPU=AMD64 -DTARGET=\"x64Win64VS2017\" -DWIN32 -D_WINDOWS -D_WIN32_WINNT=0x0501 -DWIN32_LEAN_AND_MEAN /O2 /Zi /MD /nodefaultlib:"libcmt.lib" /defaultlib:"msvcrt.lib" /EHsc -D_CRT_SECURE_NO_DEPRECATE -DNDEBUG -c
	Dynamic Release	/W3 -DSTDC99 /FS -DPtrIntType=long -DCSREAL_IS_FLOAT -DCPU=AMD64 -DTARGET=\"x64Win64VS2017\" -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 -DSTDC99 /FS -DPtrIntType=long -DCSREAL_IS_FLOAT -DCPU=AMD64 -DTARGET=\"x64Win64VS2017\" -DWIN32 -D_WINDOWS -D_WIN32_WINNT=0x0501 -DWIN32_LEAN_AND_MEAN /Od /ZI /MDd /nodefaultlib:"libcmtd.lib" /defaultlib:"msvcrtd.lib" /EHsc /RTC1 -D_CRT_SECURE_NO_DEPRECATE -c
	Dynamic Debug	/W3 -DSTDC99 /FS -DPtrIntType=long -DCSREAL_IS_FLOAT -DCPU=AMD64 -DTARGET=\"x64Win64VS2017\" -DWIN32 -D_WINDOWS -D_WIN32_WINNT=0x0501 -DWIN32_LEAN_AND_MEAN /Od /ZI /MDd /EHsc /RTC1 -D_CRT_SECURE_NO_DEPRECATE -c

13.1 Requirements when Using Microsoft Visual Studio

Note: Debug versions of applications and the various Visual C++ DLLs are not redistributable. Therefore, if you want to run debug versions, you must have the compiler installed.

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 — Redistributable Package Requirement

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

When Using Visual Studio 2015 — Update 3 Redistributable Package Requirement

You must have the Visual C++ Redistributable for Visual Studio 2015 Update 3 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 the Visual C++ Redistributable for Visual Studio 2015 Update 3 from this Microsoft website: https://www.microsoft.com/en-us/download/details.aspx?id=53840.

When Using Visual Studio 2017 — Redistributable Package Requirement

You must have the Visual C++ Redistributable for Visual Studio 2017 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 the Visual C++ Redistributable for Visual Studio 2017 from this Microsoft website: https://visualstudio.microsoft.com/vs/older-downloads/. Then look in this section: "Redistributables and Build Tools" for "Microsoft Visual C++ Redistributable for Visual Studio 2017".

When Using Visual Studio 2019

1. Redistributable Package Requirement

   You must have the Visual C++ Redistributable for Visual Studio 2019 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 the Visual C++ Redistributable for Visual Studio 2019 from this Microsoft website: https://www.visualstudio.com/downloads/. Then look in this section: "Other Tools and Frameworks" for "Microsoft Visual C++ Redistributable for Visual Studio 2019".

2. *rtiddsgen* generated projects

   Because Visual Studio 2019 is binary compatible with Visual Studio 2017, the Visual Studio 2017 libraries are used with Visual Studio 2019. There are two significant differences between the Visual Studio 2017 and Visual Studio 2019 libraries.

   a. Platform Tools Set

      The default Platform Tools set for Visual Studio 2017 is v141. The default Platform Tools set for Visual Studio 2019 is v142.

      There are two ways to upgrade the Platform Tools set. Either open the solution file with Visual Studio and follow the upgrade prompts or on the Developer Studio command line, run: **devenv /upgrade <*solutionFileName*>**

   b. Windows Target Platform Version

      The default Windows Target Platform Version for Visual Studio 2017 is 8.1. Visual Studio 2017 requires an explicit number such as 10.0.17763.0 while Visual Studio 2019 will accept a more generalized version, such as 10.0.

      There are two major ways to change the Windows Target Platform Version.

      The first way is to open the solution file in Visual Studio 2019, right-click on the solution in the 'Solution Explorer,' then select the **retarget** option in the menu.

      The second way is to edit the project file(s) with any text editor, including '**devenv /edit**', and change the tag **<WindowsTargetPlatformVersion>xx.x.xxxxx.x</WindowsTargetPlatformVersion>** to match the version of your SDK.

      RTI has introduced a workaround to the SDK versioning to simplify the process. Projects are generated with the target platform version using an environment variable like this:

      <WindowsTargetPlatformVersion>$(RTI_VS_WINDOWS_TARGET_PLATFORM_VERSION)</WindowsTargetPlatformVersion>

      Simply define **RTI_VS_WINDOWS_TARGET_PLATFORM_VERSION** to the version you have installed. One simple way you can do that is to execute this command:

      ```
      set RTI_VS_WINDOWS_TARGET_PLATFORM_VERSION=%WindowsSDKVersion:~0,-1%
      ```

      A 3rd way, which does not modify the Visual Studio 2017 projects but allows you to build with Visual Studio 2019, is to use **msbuild** to override the project settings, like this:

```
msbuild /p:PlatformToolset=v142 /p:RTI_VS_WINDOWS_TARGET_PLATFORM_
VERSION=%WindowsSDKVersion:~0,-1% <solutionFileName>
```

13.2 Linking with Libraries for Windows Platforms

Starting with *Connext DDS* 5.2.5, all *Connext DDS* libraries for Windows platforms (static release/debug, dynamic release/debug) now link with the dynamic Windows C Run-Time (CRT). Previously, the static *Connext DDS* libraries statically linked the CRT.

If you have an existing Windows project that was linking with the *Connext DDS* static libraries, you will need to change the RunTime Library settings:

- In Visual Studio, select C/C++, Code Generation, Runtime Library and use Multi-threaded DLL (**/MD**) instead of Multi-threaded (**/MT**) for static release libraries, and Multi-threaded Debug DLL (**/MDd**) instead of Multi-threaded Debug (**/MTd**) for static debug libraries.
- For command-line compilation, use **/MD** instead of **/MT** for static release libraries, and **/MDd** instead of **/MTd** for static debug libraries.

In addition, you may need to ignore the static run-time libraries in their static configurations:

- In Visual Studio, select **Linker, Input** in the project properties and add **libcmtd;libcmt** to the **'Ignore Specific Default Libraries'** entry.
- For command-line linking, add **/NODEFAULTLIB:"libcmtd" /NODEFAULTLIB:"libcmt"** to the linker options.

13.3 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.

13.4 .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.

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

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

13.5 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. Both C++03 and C++11 have been tested.

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

13.6 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.

13.7 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.)

13.8 Unsupported Features

These features are not supported on Windows platforms:

- Controlling CPU Core Affinity
- Durable Writer History and Durable Reader State
- Internal setting of thread names at the operating-system level in release mode

13.9 Monotonic Clock Support

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

13.10 Thread Configuration

See Table 13.6 Thread Settings for Windows Platforms, Table 13.7 Thread-Priority Definitions for Windows Platforms, and Table 13.8 Thread Kinds for Windows Platforms.

Table 13.6 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 13.7 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 13.8 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	

13.11 Support for 'Find Package' CMake Script

The 'Find Package' CMake script is supported on all Windows platforms in Table 13.1 Supported Windows Platforms.

For information on using this script, see Building Applications using CMake, in the RTI Connext DDS Core Libraries User's Manual.

13.12 Backtrace Support

To support the display of the backtrace on Windows systems, you need the **Dbghelp.dll** and **NtDll.dll** libraries. Without these libraries, the backtrace will not be available.

- To obtain the latest version of **DbgHelp.dll**, go to https://developer.microsoft.com/en-us/windows/downloads/windows-10-sdk and download Debugging Tools for Windows. Refer to “Calling the DbgHelp Library” for information on proper installation.

[a]See Windows manuals for additional information.

- **NtDll.dll** exports the Windows Native API. It is created automatically during the installation of the Windows operating system.

To display the backtrace information, RTI has compiled the following Windows 32-bit architectures using the **/Oy-** optimization flag to disable "Frame-Pointer Omission" optimization; you must do the same when compiling your application:

- Windows 8 - Visual Studio 2012 (Update 4) on x86 CPU (i86Win32VS2012)
- Windows 8 - Visual Studio 2013 (Update 4) on x86 CPU (i86Win32VS2013)
- Windows 10 - Visual Studio 2015 (Update 3) on x86 CPU (i86Win32VS2015)
- Windows 10 - Visual Studio 2017 (Update 2) on x86 CPU (i86Win32VS2017)

See https://docs.microsoft.com/en-us/cpp/build/reference/oy-frame-pointer-omission?view=vs-2019.

See also Logging a Backtrace for Failures, in the RTI Connext DDS Core Libraries User's Manual.

13.13 Libraries Required for Using Distributed Logger Support

RTI Distributed Logger is supported on all Windows platforms. Table 13.9 Additional Libraries for using RTI Distributed Logger lists the additional libraries you will need to use *Distributed Logger*.

Table 13.9 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

13.14 Libraries Required for Using Monitoring

To use the *RTI Secure WAN Transport* APIs, add the libraries in Table 13.10 Additional Libraries for Using Monitoring to your project files.

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 13.10 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

13.15 Libraries Required for Using RTI Secure WAN Transport APIs

To use the *RTI Secure WAN Transport* APIs, add the libraries in Table 13.11 Additional Libraries for Using RTI Secure WAN Transport APIs on Windows Systems to your project files.

Table 13.11 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	

13.16 Libraries Required for Using RTI TCP Transport APIs

To use the TCP Transport APIs, link against the additional libraries in Table 13.12 Additional Libraries for Using RTI TCP Transport APIs on Windows Systems. If you are also using *RTI TLS Support*, see Table 13.13 Additional Libraries for Using RTI TCP Transport APIs on Windows Systems with TLS Enabled. (Select the files appropriate for your chosen library format.)

---

[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.

Table 13.12 Additional Libraries for Using RTI TCP Transport APIs on Windows Systems

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

Table 13.13 Additional Libraries for Using RTI TCP Transport APIs on Windows Systems with TLS Enabled

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

13.17 Libraries Required for Zero Copy Transfer Over Shared Memory

To use the Zero Copy transfer over shared memory feature, link against the additional library in Table 13.14 Additional Libraries for Zero Copy Transfer Over Shared Memory.

Table 13.14 Additional Libraries for Zero Copy Transfer Over Shared Memory

Static Release	Static Debug	Dynamic Release	Dynamic Debug
nddsmetpz.lib	nddsmetpzd.lib	nddsmetp.dll	nddsmetpd.dll

13.18 Domain ID Support

On Windows platforms, you should avoid using ports 49152 through 65535 for inbound traffic. *Connext DDS*'s ephemeral ports (see Ports Used for Discovery, in the RTI Connext DDS Core Libraries User's

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

[b]The libraries are in **<NDDSHOME>\lib\<*architecture*>**.

Manual) may be within that range (see https://msdn.microsoft.com/en-us/library/windows/desktop/ms737550(v=vs.85).aspx).

With the default RtpsWellKnownPorts settings, port 49152 corresponds to domain ID 167, so using domain IDs 168 through 232 on Windows platforms introduces the risk of a port collision and failure to create the *DomainParticipant* when using multicast discovery. You may see this error:

```
RTIOsapiSocket_bindWithIP:OS bind() failure, error 0X271D: An attempt was made to access a
socket in a way forbidden by its access permissions.
```