RTI Data Distribution Service

The Real-Time Publish-Subscribe Middleware

Platform Notes

Version 4.5c

Printed in U.S.A. First printing.
June 2010.

Technical Support
Real-Time Innovations, Inc.
385 Moffett Park Drive
Sunnyvale, CA 94089
Phone: (408) 990-7444
Email: support@rti.com
Website: http://www.rti.com/support

Contents

Platform Notes

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

1 Supported Platforms

Table 1.1 lists the platforms available with *RTI Data Distribution Service* 4.5c.

Table 1.1 **Platforms Available with Release 4.5c**

Operating System		Reference
AIX®	AIX 5.3	Table 2.1 on page 5
INTEGRITY®	INTEGRITY 5.0	Table 3.1 on page 7
Linux® (Cell BE™)	Fedora™ 9	Table 4.1 on page 19
Linux (Intel®)	Fedora 10 Red Hat® Enterprise Linux 4.0, 5.0, 5.1 Red Hat Linux 8.0 and 9.0 SuSE® Linux Enterprise Server 10.1 (2.6 kernel)	Table 4.2 on page 19
Linux (PowerPC®)	SELinux (2.6.27.14) Yellow Dog™ Linux 4.0	Table 4.3 on page 20
LynxOS®	LynxOS 4.0, 4.2, 5.0 LynxOS-SE 3.0	Table 5.1 on page 29
Mac OS	Mac OS X	Table 6.1 on page 35
Solaris™	Solaris 2.9, 2.10	Table 7.1 on page 39

Table 1.1 **Platforms Available with Release 4.5c**

Operating System		Reference
VxWorks®	VxWorks 5.4.2, 5.5.1, 6.0 - 6.7	Table 8.1 on page 48
Windows®	Windows 2000 with Service Pack 2 or higher Windows 2003 and Windows 2003 x64 Edition Windows CE 6.0 Windows Vista® and Windows Vista x64 Edition Windows XP Professional and Windows XP Professional x64 Edition	Table 9.1 on page 61

For each platform, this document provides information on:

- ❏ Supported operating systems and compilers
- ❏ Required *RTI Data Distribution Service* and system libraries
- ❏ Required compiler and linker flags
- ❏ Required environment variables for running the application (if any)
- ❏ Details on how the *RTI Data Distribution Service* libraries were built
- ❏ Multicast support
- ❏ Supported transports
- ❏ Monotonic clock support
- ❏ CPU core affinity control support

2 AIX Platforms

Table 2.1 on page 5 lists the architectures supported on the IBM® AIX operating system.

Table 2.2 on page 5 lists the compiler flags and the libraries you will need to link into your application.

Table 2.3 on page 6 provides details on the environment variables required to be set at run time for an AIX architecture.

Table 2.4 on page 6 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.

2.1 Changing Thread Priority

Due to the AIX threading-model implementation, there are situations that require you to run your *RTI Data Distribution Service* 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 and 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 and DDS_DomainParticipantQos.receiver_pool.thread).

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 online documentation for more information.

2.3 Supported 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 *RTI Data Distribution Service* 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 *RTI Data Distribution Service* 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 *RTI Data Distribution Service* are in the range of 0x400000. For example:

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

The shared semaphore keys used by *RTI Data Distribution Service* 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 *RTI Data Distribution Service*.

2.4 Monotonic Clock Support

The monotonic clock (described in Section 8.6 in the *RTI Data Distribution Service User's Manual*) is not supported on AIX platforms.

2.5 Support for Controlling CPU Core Affinity for RTI Threads

Support for controlling CPU core affinity (described in Section 17.5 in the *RTI Data Distribution Service User's Manual*) is not available for AIX platforms.

Table 2.1 **Supported AIX Target Platforms**

Operating System	CPU	Compiler	RTI Architecture Abbreviation
AIX 5.3	POWER5 (32-bit mode)	IBM XLC for AIX v9.0	p5AIX5.3xlc9.0
		IBM Java 1.6	p5AIX5.3xlc9.0jdk
	POWER5 (64-bit mode)	IBM XLC for AIX v9.0	64p5AIX5.3xlc9.0
		IBM Java 1.6	64p5AIX5.3xlc9.0jdk

Table 2.2 **Building Instructions for AIX Architectures**

API	Library Format	Required RTI Libraries[a]	Required System Libraries[b]	Required Compiler Flags
C++	Static Release	libnddscppz.a libnddscz.a libnddscorez.a	-ldl -lnsl -lm -pthread	-DRTI_AIX -DRTI_UNIX -q[32|64] [c] -qlongdouble
	Static Debug	libnddscppzd.a libnddsczd.a libnddscorezd.a		
	Dynamic Release	libnddscpp.so libnddsc.so libnddscore.so	-ldl -lnsl -lm -pthread -brtl	
	Dynamic Debug	libnddscppd.so libnddscd.so libnddscored.so		
C	Static Release	libnddscz.a libnddscorez.a	-ldl -lnsl -lm -pthread	-DRTI_AIX -DRTI_UNIX -q[32|64] [c] -qlongdouble -qthreaded [d]
	Static Debug	libnddsczd.a libnddscorezd.a		
	Dynamic Release	libnddsc.so libnddscore.so	-ldl -lnsl -lm -pthread -brtl	
	Dynamic Debug	libnddscd.so libnddscored.so		
Java	Release	nddsjava.jar	N/A	N/A
	Debug	nddsjavad.jar		

a. The *RTI Data Distribution Service* C/C++ libraries are located in $(NDDSHOME)/lib/<*architecture*>/.
(where $(NDDSHOME) is where *RTI Data Distribution Service* is installed, such as /local/rti/ndds.4.5x)

b. Transports (other than the default IP transport) such as StarFabric may require linking in additional libraries. For further details, see the online documentation or contact support@rti.com.

c. Use '-q32' if you build 32-bit code or '-q64' for 64-bit code.

d. 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 and Debug)	Required Environment Variables
p5AIX5.3xlc9.0jdk 64p5AIX5.3xlc9.0jdk	N/A	LD_LIBRARY_PATH=$(NDDSHOME)/lib/<*arch*>: $(LD_LIBRARY_PATH) EXTSHM=ON[a]
All other supported architectures	Static	EXTSHM=ON[a]
	Dynamic	LD_LIBRARY_PATH=$(NDDSHOME)/lib/<*arch*>: $(LD_LIBRARY_PATH) EXTSHM=ON[a]

a. See Notes for Using Shared Memory (Section 2.3.1).

Table 2.4 **Library-Creation Details for AIX Architectures**

RTI Architecture	Library Format (Static & Dynamic)	Compiler Flags Used by RTI [a]
p5AIX5.3xlc9.0jdk	Release	-q32 -qlongdouble -qalias=noansi -qpic=large -qthreaded -D_POSIX_C_SOURCE=199506L -D__EXTENSIONS__ -O -qflag=i:i -DPtrIntType=long -DCSREAL_IS_FLOAT -DCPU=Power5+ -DNDEBUG
	Debug	-q32 -qlongdouble -qalias=noansi -qpic=large -qthreaded -D_POSIX_C_SOURCE=199506L -D__EXTENSIONS__ -O -qflag=i:i -DPtrIntType=long -DCSREAL_IS_FLOAT -DCPU=Power5+ -g
64p5AIX5.3xlc9.0	Release	-q64 -qwarn64 -qlongdouble -qalias=noansi -qpic=large -qthreaded -D_POSIX_C_SOURCE=199506L -D__EXTENSIONS__ -O -qflag=i:i -DPtrIntType=long -DCSREAL_IS_FLOAT -DCPU=Power5+ -DNDEBUG
	Debug	-q64 -qwarn64 -qlongdouble -qalias=noansi -qpic=large -qthreaded -D_POSIX_C_SOURCE=199506L -D__EXTENSIONS__ -O -qflag=i:i -DPtrIntType=long -DCSREAL_IS_FLOAT -DCPU=Power5+ -g

Table 2.4 **Library-Creation Details for AIX Architectures**

RTI Architecture	Library Format (Static & Dynamic)	Compiler Flags Used by RTI [a]
p5AIX5.3xlc9.0jdk	Release	-target 1.4 -source 1.4
	Debug	-target 1.4 -source 1.4 -g
64p5AIX5.3xlc9.0jdk	Release	-target 1.6 -source 1.6
	Debug	-target 1.6 -source 1.6 -g

a. *RTI Data Distribution Service* 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**.

3 INTEGRITY Platforms

Table 3.1 lists the architectures supported on the INTEGRITY® operating system.

Table 3.1 **Supported INTEGRITY Target Platforms**[a]

Operating System	CPU	IP Stack	RTI Architecture Abbreviation
INTEGRITY 5.0.7	PPC 74XX	InterNiche (GHnet1) TCP/IP stack	ppc7400Inty5.0.7.mvme5100-7400 [c]
		Interpeak TCP/IP stack with multicast	ppc7400Inty5.0.7.mvme5100-7400-ipk[b]
INTEGRITY 5.0.8	PPC 74XX	InterNiche (GHnet1) TCP/IP stack	ppc7400Inty5.0.7.mvme5100-7400 [c]
INTEGRITY 5.0.9, 5.0.10	PPC 74XX	GHnet2 TCP/IP stack	ppc7400Inty5.0.9.mvme5100-7400-ghnet2

a. For use with Windows and Solaris hosts, as supported by Green Hills Systems.

b. The RTI library for ppc7400Inty5.0.7.mvme5100-7400-ipk is compatible with both IPLITE and IPNET versions of the Interpeak stack (version 6). To successfully use IPLITE (resp IPNET) with *RTI Data Distribution Service*, you must recompile libipsocket with the -DIPLITE option (resp -DIPNET).

c. INTEGRITY 5.0.7 and 5.0.8 share the same architecture (ppc7400Inty5.0.7.mvme5100-7400).

Table 3.1 on page 14 lists the compiler flags and the libraries you will need to link into your application.

Table 3.2 on page 14 provides details on the environment variables required to be set at run time for an INTEGRITY architecture.

Table 3.3 on page 15 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.

3.1 Diagnostics on INTEGRITY Systems

RTI Data Distribution Service 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.*

3.2 Socket-Enabled and POSIX-Enabled Threads are Required

RTI Data Distribution Service on INTEGRITY platforms internally relies on the POSIX API for many of its system calls. As a result, any thread calling *RTI Data Distribution Service* 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 *RTI Data Distribution Service* 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 *RTI Data Distribution Service* must be socket-enabled. This can be achieved by calling **InitLibSocket()** before making any *RTI Data Distribution Service* 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.

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

RTI Data Distribution Service can run on all four CPUs, provided the following hold true:

- ❏ *RTI Data Distribution Service* 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. *RTI Data Distribution Service* has been tested with the following libraries built into the INTEGRITY kernel: **debug**, **res**, **load**, **socket**, **itcpip**, **lbp**, **queue**, **ifbp**, **idb**, **bsl**.

3.4 Multi-NIC Support on INTEGRITY 5.0

Due to limitations with the API of the InterPeak stack for INTEGRITY 5.0, *RTI Data Distribution Service* 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.

3.5 Multicast Support

Multicast is supported on all INTEGRITY 5 platforms (InterNiche, Interpeak, and GHnet2).

3.6 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.

3.6.1 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 8 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.

3.6.2 Using Shared Memory on INTEGRITY 5.0

RTI Data Distribution Service uses the single address-space POSIX library to implement the shared-memory transport on the INTEGRITY 5.0 operating system.

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 *RTI Data Distribution Service* 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 *RTI Data Distribution Service* application(s), and another one to contain the **posix_shm_manager**.

RTI Data Distribution Service 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 this according to the path to your INTEGRITY distribution):

   ```
   -os_dir=/local/applications/integrity/integrity-5.0.4
   ```

2. Build the project.
3. Before running your *RTI Data Distribution Service* 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). *RTI Data Distribution Service* uses the first one, but will work if you are using the complete POSIX implementation.

3.6.3 Shared Memory Limitations on INTEGRITY 5.0

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 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 *RTI Data Distribution Service* 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 *RTI Data Distribution Service*. 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.

3.7 Using rtiddsping and rtiddsspy on PowerPC INTEGRITY Systems

While the RTI libraries for INTEGRITY can be used with any BSP, providing the PowerPC processor falls under the same category (for example, the ppc7400... RTI libraries can be used on any target with a PPC74xx processor), *rtiddsping* and *rtiddsspy* are provided as executables, and therefore are BSP-dependent. You will not be able to run them successfully on your target if it is not compatible with the BSP listed in the architecture name (such as mvme5100-7400). Please refer to your hardware documentation for peripheral compatibility across BSPs.

3.8 Monotonic Clock Support

The monotonic clock (described in Section 8.6 in the *RTI Data Distribution Service User's Manual*) is not supported.

3.9 Support for Controlling CPU Core Affinity for RTI Threads

Support for controlling CPU core affinity (described in Section 17.5 in the *RTI Data Distribution Service User's Manual*) is not available for INTEGRITY platforms.

3.10 Issues with INTEGRITY Systems

3.10.1 Ungraceful Shutdown When Using C++ and Dynamic System Libraries on INTEGRITY 5.0.7 Systems

Due to an issue with the INTEGRITY 5.0.7 tool chain, C++ applications that are linked against shared system libraries and include **stdio.h** may encounter an exception at shutdown.

The suggested workaround from Green Hills is to link against the static versions of the system libraries. In the case of the *rtiddsgen*-generated INTEGRITY project files, you may do so by following these steps:

1. Add the following line to your top-level project file, **<IDL file name>_default.gpj**:

   ```
   -non_shared
   ```

2. Remove all the shared objects from the three **.int** files in your project (**<IDL file name>_publisherdd.int**, **<IDLfilename>_subscriberdd.int**, **posix_shm_manager.int**) by removing the following lines:

   ```
   Library         libINTEGRITY.so
   Library         libc.so
   Library         libscxx.so
   ```

3. Rebuild your top-level project, **<IDL file name>_default.gpj**.

3.10.2 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 Bug # 10768]

3.10.3 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.

3.10.4 Compiler Warnings Regarding Unrecognized #pragma Directives

Building *RTI Data Distribution Service* projects for INTEGRITY causes the compiler to produce several warnings about #pragma directives not recognized in some *RTI Data Distribution Service* 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.

3.10.5 Warning when Loading RTI Data Distribution Service Applications on INTEGRITY 5.0.x Systems

When an *RTI Data Distribution Service* 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 cancella-
tion will not work."
```

The *RTI Data Distribution Service* 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 *RTI Data Distribution Service* 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").

Table 3.1 **Building Instructions for INTEGRITY Architectures**

API	Library Format	Required RTI Libraries[a]	Required System Libraries[b]	Required Compiler Flags
C++	Static Release	libnddscppz.a libnddscz.a libnddscorez.a	libsocket.a libnet.a libposix.a	RTI_INTY
	Static Debug	libnddscppzd.a libnddsczd.a libnddscorezd.a (libnddscppzd.dba)[c] (libnddsczd.dba)[c] (libnddscorezd.dba)[c]		
C	Static Release	libnddscz.a libnddscorez.a		
	Static Debug	libnddsczd.a libnddscorezd.a (libnddsczd.dba)[c] (libnddscorezd.dba)[c]		

a. The *RTI Data Distribution Service* C/C++ libraries are located in $(NDDSHOME)/lib/<*architecture*>/.
(where $(NDDSHOME) is where *RTI Data Distribution Service* is installed, such as /local/rti/ndds.4.5x)

b. Transports (other than the default IP transport) such as StarFabric may require linking in additional libraries. For further details, see the online documentation or contact support@rti.com.

c. 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).

Table 3.2 **Running Instructions for INTEGRITY Architectures**

RTI Architecture	Required Environment Variables
All INTEGRITY architectures	None

Table 3.3 **Library-Creation Details for INTEGRITY Architectures**

RTI Architecture	Library Format	Compiler Flags Used by RTI
ppc7400Inty5.0.7.mvme5100-7400	Static Release	-bspname=mvme5100-7400 -prefixed_msgs --unknown_pragma_silent -O -DRTS_INTY -DPtrIntType=long -DCSREAL_IS_FLOAT -DCPU= -DTARGET=\"ppc7400Inty5.0.7.mvme5100-7400\" -DNDEBUG -c
	Static Debug	-bspname=mvme5100-7400 -prefixed_msgs --unknown_pragma_silent -G -DRTS_INTY -DPtrIntType=long -DCSREAL_IS_FLOAT -DCPU= -DTARGET=\"ppc7400Inty5.0.7.mvme5100-7400\" -c
ppc7400Inty5.0.7.mvme5100-7400-ipk	Static Release	-bspname=mvme5100-7400 -prefixed_msgs --unknown_pragma_silent -O -DRTS_INTY -DPtrIntType=long -DCSREAL_IS_FLOAT -DCPU= -DTARGET=\"ppc7400Inty5.0.7.mvme5100-7400-ipk\" -DNDEBUG -c
	Static Debug	-bspname=mvme5100-7400 -prefixed_msgs --unknown_pragma_silent -G -DRTS_INTY -DPtrIntType=long -DCSREAL_IS_FLOAT -DCPU= -DTARGET=\"ppc7400Inty5.0.7.mvme5100-7400-ipk\" -c
ppc7400Inty5.0.9.mvme5100-7400-ghnet2	Static Release	-bspname=mvme5100-7400 -prefixed_msgs --unknown_pragma_silent -G -DRTS_INTY -DPtrIntType=long -DCSREAL_IS_FLOAT -DCPU= -DTARGET=\"ppc7400Inty5.0.9.mvme5100-7400\" -c
	Static Debug	

4 Linux and Fedora Platforms

First, see the basic instructions for compiling on Linux platforms provided in Section 9.3 in the *RTI Data Distribution Service User's Manual*. The following tables provide supplemental information.

Table 4.1 on page 19 through Table 4.3 on page 20 list the supported Linux and Fedora architectures.

Table 4.4 on page 21 lists the compiler flags and libraries you will need to link into your application. Do not mix libraries built for different formats. (See also: Monotonic Clock Support (Section 4.4).)

Note: Unlike the other Linux architectures, Yellow Dog Linux (ppc7400Linux2.6gcc3.3.3) is only supported as a *target*, not a host; therefore you must run *rtiddsgen* and build the application on a separate computer.

Table 4.5 on page 22 provides details on the environment variables required to be set at run time for a Linux architecture. When running on 64-bit Java architectures (x64Linux2.4...jdk), use the **-d64** flag in the command-line.

Table 4.6 on page 22 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 4.7 on page 26 and Table 4.8 on page 26 list additional libraries required when using the optional *RTI Secure WAN Transport* and *RTI TCP Transport*, respectively.

4.1 Native POSIX Thread Library (NPTL) Requirements

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

- Red Hat Enterprise Linux 4.0: i86Linux2.6gcc3.4.3 and x64Linux2.6gcc3.4.5
- Yellow Dog Linux 4.0: ppc7400Linux2.6gcc3.3.3

This section applies only to the above architectures.

- When you *build* the application, you must have the development NPTL library installed in **/usr/lib/nptl**. This library is not installed by default.
- To see if your system has NPTL installed, look for this directory: **/usr/lib/nptl**. It should contain these files: **libpthread.so** and **libpthread.a**.

If NPTL not installed, you will need to install a package that includes it, such as **nptl-devel**. This package is not typically part of a default installation. You can find it either in your original Linux installation media (CD/DVD) or, if you have upgraded your system, through the distribution's update site.

- ❑ When you *run* the application, it will automatically use the default NPTL library in **/lib/nptl**. You do not need the development library installed on the target system.

Note: Make sure the environment variable LD_ASSUME_KERNEL is either not defined at all, or is set to 2.4.20 or higher. The middleware will not run if it is set to less than 2.4.20.[1]

4.2 Multicast Support

Multicast is supported on all Linux and Fedora 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.

Note—Group Address Ignored for Multicast Reception on Loopback: On Linux and Fedora architectures, the multicast-loopback implementation ignores the group address when receiving messages. This causes *RTI Data Distribution Service* 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 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.)

4.3 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 Linux and Fedora platforms except Red Hat 8 (i86Linux2.4gcc3.2.2) and Red Hat 9 (i86Linux2.4gcc3.2). The transport is not enabled by default, and the peers list must be modified to support IPv6. Note that mapping of the TransportPriority QoS is not supported.

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

TCP/IPv4: Supported on Red Hat Enterprise Linux 4.0 and higher. (This is *not* a built-in transport.)

4.3.1 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 *RTI Data Distribution Service* are in the range of 0x400000. For example:

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

The shared semaphore keys used by *RTI Data Distribution Service* 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 *RTI Data Distribution Service*.

4.4 Monotonic Clock Support

The monotonic clock (described in Section 8.6 in the *RTI Data Distribution Service User's Manual*) is supported on all Linux and Fedora 2.6 kernel platforms.

4.5 Support for Controlling CPU Core Affinity for RTI Threads

Support for controlling CPU core affinity (described in Section 17.5 in the *RTI Data Distribution Service User's Manual*) is available on all supported Linux, SuSE, and Fedora platforms except those with "Linux2.4" in the architecture name (that is, those running on a 2.4 kernel).

Note: The API for controlling CPU core affinity may change in future releases.

4.6 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 *RTI Data Distribution Service* 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 Release Notes Installation Guide* for details.

To use the WAN or Secure Transport APIs, link against the additional libraries from Table 4.7 on page 26. (Select the files appropriate for your chosen library format.)

4.7 Libraries Required for Using RTI TCP Transport APIs

To use the TCP Transport APIs, link against the additional libraries from Table 4.8 on page 26. If you are using *RTI TLS Support*, see Table 4.9 on page 26. (Select the files appropriate for your chosen library format.)

Table 4.1 **Linux Platforms on Cell BE CPUs**

Operating System	CPU	Compiler	RTI Architecture Abbreviation
Fedora 9 (2.6 kernel)	64-bit cell	ppu 4.1.1	cell64Linux2.6ppu4.1.1
		Sun Java Platform Standard Edition JDK 1.5 and 1.6	cell64Linux2.6ppu4.1.1jdk

Table 4.2 **Linux Platforms on Intel and AMD CPUs**

Operating System	CPU	Compiler	RTI Architecture Abbreviation
Fedora 10 (2.6 kernel)	x64	gcc 4.3.2	x64Linux2.6gcc4.3.2
		Sun Java Platform Standard Edition JDK 1.6	x64Linux2.6gcc4.3.2jdk
Red Hat Linux 8.0 (2.4 kernel)	Pentium class	gcc 3.2	i86Linux2.4gcc3.2
		Sun Java Platform Standard Edition JDK 1.5 and 1.6	i86Linux2.4gcc3.2jdk
Red Hat Linux 9.0 (2.4 kernel)	Pentium class	gcc 3.2.2	i86Linux2.4gcc3.2.2[a]
		Sun Java Platform Standard Edition JDK 1.5 and 1.6	i86Linux2.4gcc3.2.2jdk
Red Hat Enterprise Linux 3.0 (2.4 kernel)	Pentium class	gcc 3.2.2	i86Linux2.4gcc3.2.2[a]
		Sun Java Platform Standard Edition JDK 1.5 and 1.6	i86Linux2.4gcc3.2.2jdk
	AMD64	gcc 3.2.3	x64Linux2.4gcc3.2.3
		Sun Java Platform Standard Edition JDK 1.5 and 1.6	x64Linux2.4gcc3.2.3jdk

Table 4.2 **Linux Platforms on Intel and AMD CPUs**

Operating System	CPU	Compiler	RTI Architecture Abbreviation
Red Hat Enterprise Linux 4.0 (2.6 kernel)	Pentium class	gcc 3.4.3	i86Linux2.6gcc3.4.3
		Sun Java Platform Standard Edition JDK 1.5 and 1.6	i86Linux2.6gcc3.4.3jdk
	x86_64 and AMD64	gcc 3.4.5	x64Linux2.6gcc3.4.5
		Sun Java Platform Standard Edition JDK 1.5 and 1.6	x64Linux2.6gcc3.4.5jdk
Red Hat Enterprise Linux 5.0 (2.6 kernel)	Pentium class	gcc 4.1.1	i86Linux2.6gcc4.1.1
		Sun Java Platform Standard Edition JDK 1.5 and 1.6	i86Linux2.6gcc4.1.1jdk
	x86_64 and AMD64	gcc 4.1.1	x64Linux2.6gcc4.1.1
		Sun Java Platform Standard Edition JDK 1.5 and 1.6	x64Linux2.6gcc4.1.1jdk
Red Hat Enterprise Linux 5.1, 5.2 (2.6 kernel)	Pentium class	gcc 4.1.2	i86Linux2.6gcc4.1.2
		Sun Java Platform Standard Edition JDK 1.6	i86Linux2.6gcc4.1.2jdk
	x86_64 and AMD64	gcc 4.1.2	x64Linux2.6gcc4.1.2
		Sun Java Platform Standard Edition JDK 1.6	x64Linux2.6gcc4.1.2jdk
SuSE Linux Enterprise Server 10.1 (2.6 kernel)	Pentium class	gcc 4.1.0	i86Suse10.1gcc4.1.0
		Sun Java Platform Standard Edition JDK 1.5 and 1.6	i86Suse10.1gcc4.1.0jdk
	AMD64	gcc 4.1.0	x64Suse10.1gcc4.1.0
		Sun Java Platform Standard Edition JDK 1.5 and 1.6	x64Suse10.1gcc4.1.0jdk

a. Red Hat Linux 9.0 and Red Hat Enterprise Linux 3.0 share the same RTI architecture.

Table 4.3 **Linux Platforms on PowerPC CPUs**

Operating System	CPU	Compiler	RTI Architecture Abbreviation
SELinux (2.6.27.14)	PowerPC440EP	gcc 4.3.3 with GNU libc 2.9	ppc4xxFPLinux2.6gcc4.3.3
Yellow Dog® Linux 4.0 (2.6 kernel) (target only)	PPC 74xx (such as 7410)	gcc 3.3.3	ppc7400Linux2.6gcc3.3.3

Table 4.4 **Building Instructions for Linux and Fedora Architectures**

API	Library Format	Required RTI Libraries or Jar Files[a]	Required System Libraries	Required Compiler Flags
C++	Static Release	libnddscppz.a libnddscz.a libnddscorez.a	For all *Linux2.6gcc3* architectures: -ldl -lnsl -lm -L/usr/lib/nptl -lpthread -lrt All other Linux architectures: -ldl -lnsl -lm -lpthread -lrt	-DRTI_UNIX If using a 64-bit architecture, this flag is also required: -m64 If using a 32-bit architecture, this flag is also required: -m32
	Static Debug	libnddscppzd.a libnddsczd.a libnddscorezd.a		
	Dynamic Release	libnddscpp.so libnddsc.so libnddscore.so		
	Dynamic Debug	libnddscppd.so libnddscd.so libnddscored.so		
C	Static Release	libnddscz.a libnddscorez.a	For all *Linux2.6gcc3* architectures: -ldl -lnsl -lm -L/usr/lib/nptl -lpthread -lrt All other Linux architectures: -ldl -lnsl -lm -lpthread -lrt	-DRTI_UNIX If using a 64-bit architecture, this flag is also required: -m64 If using a 32-bit architecture, this flag is also required: -m32
	Static Debug	libnddsczd.a libnddscorezd.a		
	Dynamic Release	libnddsc.so libnddscore.so		
	Dynamic Debug	libnddscd.so libnddscored.so		
Java	Release	nddsjava.jar	N/A	None required
	Debug	nddsjavad.jar		

a. RTI C/C++ libraries are in $(NDDSHOME)/lib/<*architecture*>/. RTI Java files are in $(NDDSHOME)/class/ (where $(NDDSHOME) is where *RTI Data Distribution Service* is installed, such as /local/rti/ndds.4.5x).

Table 4.5 **Running Instructions for Linux and Fedora Architectures**

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

a. ${NDDSHOME} represents the root directory of your *RTI Data Distribution Service* installation. ${LD_LIBRARY_PATH} represents the value of the LD_LIBRARY_PATH variable prior to changing it to support *RTI Data Distribution Service*. 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 4.6 **Library-Creation Details for Linux and Fedora Architectures**

RTI Architecture	Library Format *(Static and Dynamic)*	Compiler Flags Used by RTI
cell64Linux2.6ppu4.1.1	Release	-m64 -O3 -fPIC -DLINUX -Wall -Wno-unknown-pragmas -DRTS_UNIX -DPtrIntType=long -DCSREAL_IS_FLOAT -DCPU=cell -DTARGET=\"cell64Linux2.6ppu4.1.1\" -DNDEBUG -c -Wp,-MD
	Debug	-m64 -O3 -fPIC -DLINUX -g -Wall -Wno-unknown-pragmas -DRTS_UNIX -DPtrIntType=long -DCSREAL_IS_FLOAT -DCPU=cell -DTARGET=\"cell64Linux2.6ppu4.1.1\" -c -Wp,-MD
i86Linux2.4gcc3.2[a]	Release	-fPIC -DLINUX -O -Wall -Wno-unknown-pragmas -DRTS_UNIX -DPtrIntType=long -DCPU=I80586 -DTARGET=\"i86Linux2.4gcc3.2\" -DNDEBUG -c -Wp,-MD
	Debug	-fPIC -DLINUX -g -Wall -Wno-unknown-pragmas -DRTS_UNIX -DPtrIntType=long -DCPU=I80586 -DTARGET=\"i86Linux2.4gcc3.2\" -c -Wp,-MD
i86Linux2.4gcc3.2.2[a]	Release	-fPIC -DLINUX -O -Wall -Wno-unknown-pragmas -DRTS_UNIX -DPtrIntType=long -DCPU=I80586 -DTARGET=\"i86Linux2.4gcc3.2.2\" -DNDEBUG -c -Wp,-MD
	Debug	-fPIC -DLINUX -g -Wall -Wno-unknown-pragmas -DRTS_UNIX -DPtrIntType=long -DCPU=I80586 -DTARGET=\"i86Linux2.4gcc3.2.2\" -c -Wp,-MD

Table 4.6 **Library-Creation Details for Linux and Fedora Architectures**

RTI Architecture	Library Format (*Static and Dynamic*)	Compiler Flags Used by RTI
i86Linux2.6gcc3.4.3[a]	Release	-fPIC -DLINUX -O -Wall -Wno-unknown-pragmas -DRTS_UNIX -DPtrIntType=long -DCSREAL_IS_FLOAT -DCPU=I80586 -DTARGET=\"i86Linux2.6gcc3.4.3\" -fmessage-length=0 -DNDEBUG -c -Wp,-MD
	Debug	-fPIC -DLINUX -g -Wall -Wno-unknown-pragmas -DRTS_UNIX -DPtrIntType=long -DCSREAL_IS_FLOAT -DCPU=I80586 -DTARGET=\"i86Linux2.6gcc3.4.3\" -fmessage-length=0 -c -Wp,-MD
i86Linux2.6gcc4.1.1	Release	-fPIC -DLINUX -O -Wall -Wno-unknown-pragmas -DRTS_UNIX -DPtrIntType=long -DCSREAL_IS_FLOAT -DCPU=I80586 -DTARGET=\"i86Linux2.6gcc4.1.1\" -fmessage-length=0 -DNDEBUG -c -Wp,-MD
	Debug	-fPIC -DLINUX -g -Wall -Wno-unknown-pragmas -DRTS_UNIX -DPtrIntType=long -DCSREAL_IS_FLOAT -DCPU=I80586 -DTARGET=\"i86Linux2.6gcc4.1.1\" -fmessage-length=0 -c -Wp,-MD
i86Linux2.6gcc4.1.2	Release	-fPIC -DLINUX -O -Wall -Wno-unknown-pragmas -DRTS_UNIX -DPtrIntType=long -DCSREAL_IS_FLOAT -DCPU=I80586 -DTARGET=\"i86Linux2.6gcc4.1.2\" -fmessage-length=0 -DNDEBUG -c -Wp,-MD
	Debug	-fPIC -DLINUX -g -Wall -Wno-unknown-pragmas -DRTS_UNIX -DPtrIntType=long -DCSREAL_IS_FLOAT -DCPU=I80586 -DTARGET=\"i86Linux2.6gcc4.1.2\" -fmessage-length=0 -c -Wp,-MD
i86Suse10.1gcc4.1.0	Release	-fPIC -DLINUX -O -Wall -Wno-unknown-pragmas -DRTS_UNIX -DPtrIntType=long -DCSREAL_IS_FLOAT -DCPU=I80586 -DTARGET=\"i86Suse10.1gcc4.1.0\" -fmessage-length=0 -DNDEBUG -c -Wp,-MD
	Debug	-fPIC -DLINUX -g -Wall -Wno-unknown-pragmas -DRTS_UNIX -DPtrIntType=long -DCSREAL_IS_FLOAT -DCPU=I80586 -DTARGET=\"i86Suse10.1gcc4.1.0\" -fmessage-length=0 -c -Wp,-MD
ppc7400Linux2.6gcc3.3.3[a]	Release	-fPIC -DLINUX -O -Wall -Wno-unknown-pragmas -DRTS_UNIX -DPtrIntType=long -DCSREAL_IS_FLOAT -DCPU=PPC7400 -DTARGET=\"ppc7400Linux2.6gcc3.3.3\" -DNDEBUG -c -Wp,-MD
	Debug	-fPIC -DLINUX -g -Wall -Wno-unknown-pragmas -DRTS_UNIX -DPtrIntType=long -DCSREAL_IS_FLOAT -DCPU=PPC7400 -DTARGET=\"ppc7400Linux2.6gcc3.3.3\" -c -Wp,-MD

Table 4.6 **Library-Creation Details for Linux and Fedora Architectures**

RTI Architecture	Library Format (*Static and Dynamic*)	Compiler Flags Used by RTI
ppc4xxFPLinux2.6gcc4.3.3	Release	-fPIC -DLINUX -O -Wall -Wno-unknown-pragmas -DRTS_UNIX -DPtrIntType=long -DCSREAL_IS_FLOAT -DCPU=4xxFP -DTARGET=\"ppc4xxFPLinux2.6gcc4.3.3\" -DNDEBUG -c -Wp,-MD
	Debug	-fPIC -DLINUX -g -Wall -Wno-unknown-pragmas -DRTS_UNIX -DPtrIntType=long -DCSREAL_IS_FLOAT -DCPU=4xxFP -DTARGET=\"ppc4xxFPLinux2.6gcc4.3.3\" -DNDEBUG -c -Wp,-MD
x64Linux2.4gcc3.2.3[a]	Release	-m64 -fPIC -DLINUX -O -Wall -Wno-unknown-pragmas -DRTS_UNIX -DPtrIntType=long -DCSREAL_IS_FLOAT -DCPU=AMD64 -DTARGET=\"x64Linux2.6gcc3.2.3\" -DNDEBUG -fmessage-length=0 -c -Wp,-MD
	Debug	-m64 -fPIC -DLINUX -g -Wall -Wno-unknown-pragmas -DRTS_UNIX -DPtrIntType=long -DCSREAL_IS_FLOAT -DCPU=AMD64 -DTARGET=\"x64Linux2.4gcc3.2.3\" -fmessage-length=0 -c -Wp,-MD
x64Linux2.6gcc3.4.5[a]	Release	-m64 -fPIC -DLINUX -O -Wall -Wno-unknown-pragmas -DRTS_UNIX -DPtrIntType=long -DCSREAL_IS_FLOAT -DCPU=AMD64 -DTARGET=\"x64Linux2.6gcc3.4.5\" -DNDEBUG -c -Wp,-MD
	Debug	-m64 -fPIC -DLINUX -g -Wall -Wno-unknown-pragmas -DRTS_UNIX -DPtrIntType=long -DCSREAL_IS_FLOAT -DCPU=AMD64 -DTARGET=\"x64Linux2.6gcc3.4.5\" -fmessage-length=0 -c -Wp,-MD
x64Linux2.6gcc4.1.1[a]	Release	-m64 -fPIC -DLINUX -O -Wall -Wno-unknown-pragmas -DRTS_UNIX -DPtrIntType=long -DCSREAL_IS_FLOAT -DCPU=AMD64 -DTARGET=\"x64Linux2.6gcc4.1.1\" -DNDEBUG -c -Wp,-MD
	Debug	-m64 -fPIC -DLINUX -g -Wall -Wno-unknown-pragmas -DRTS_UNIX -DPtrIntType=long -DCSREAL_IS_FLOAT -DCPU=AMD64 -DTARGET=\"x64Linux2.6gcc4.1.1\" -fmessage-length=0 -c -Wp,-MD
x64Linux2.6gcc4.1.2[a]	Release	-m64 -fPIC -DLINUX -O -Wall -Wno-unknown-pragmas -DRTS_UNIX -DPtrIntType=long -DCSREAL_IS_FLOAT -DCPU=AMD64 -DTARGET=\"x64Linux2.6gcc4.1.2\" -DNDEBUG -c -Wp,-MD
	Debug	-m64 -fPIC -DLINUX -g -Wall -Wno-unknown-pragmas -DRTS_UNIX -DPtrIntType=long -DCSREAL_IS_FLOAT -DCPU=AMD64 -DTARGET=\"x64Linux2.6gcc4.1.2\" -fmessage-length=0 -c -Wp,-MD

Table 4.6 **Library-Creation Details for Linux and Fedora Architectures**

RTI Architecture	Library Format (*Static and Dynamic*)	Compiler Flags Used by RTI
x64Linux2.6gcc4.3.2[a]	Release	-m64 -fPIC -DLINUX -O -Wall -Wno-unknown-pragmas -DRTS_UNIX -DPtrIntType=long -DCSREAL_IS_FLOAT -DCPU=AMD64 -DTARGET=\"x64Linux2.6gcc4.3.2\" -DNDEBUG -c -Wp,-MD
	Debug	-m64 -fPIC -DLINUX -g -Wall -Wno-unknown-pragmas -DRTS_UNIX -DPtrIntType=long -DCSREAL_IS_FLOAT -DCPU=AMD64 -DTARGET=\"x64Linux2.6gcc4.3.2\" -c -Wp,-MD
x64Suse10.1gcc4.1.0	Release	-m64 -fPIC -DLINUX -O -Wall -Wno-unknown-pragmas -DRTS_UNIX -DPtrIntType=long -DCSREAL_IS_FLOAT -DCPU=AMD64 -DTARGET=\"x64Suse10.1gcc4.1.0\" -DNDEBUG -c -Wp,-MD
	Debug	-m64 -fPIC -DLINUX -g -Wall -Wno-unknown-pragmas -DRTS_UNIX -DPtrIntType=long -DCSREAL_IS_FLOAT -DCPU=AMD64 -DTARGET=\"x64Suse10.1gcc4.1.0\" -fmessage-length=0 -c -Wp,-MD
i86Linux2.4gcc3.2jdk i86Linux2.4gcc3.2.2jdk i86Linux2.6gcc3.4.3jdk i86Linux2.6gcc4.1.1jdk i86Linux2.6gcc4.1.2jdk i86Suse10.1gcc4.1.0jdk	Dynamic Release	-target 1.4 –source 1.4
	Dynamic Debug	-target 1.4 –source 1.4 -g
cell64Linux2.6ppu4.1.1jdk x64Linux2.4gcc3.2.3jdk x64Linux2.6gcc3.4.5jdk x64Linux2.6gcc4.1.1jdk x64Linux2.6gcc4.1.2jdk x64Linux2.6gcc4.3.2jdk x64Suse10.1gcc4.1.0jdk	Dynamic Release	-target 1.6 -source 1.6
	Dynamic Debug	-target 1.6 -source 1.6 -g

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

Table 4.7 **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	

a. The libraries are located in <*wan install dir*>/lib/<*architecture*>/., where <*wan install dir*> is where you installed *RTI Secure WAN Transport*, such as /local/rti/ndds.4.5x.

b. These libraries are located <*openssl install dir*>/<*architecture*/lib, where <*openssl install dir*> is where you installed OpenSSL, such as /local/rti/openssl-0.9.8f.

Table 4.8 **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

a. The libraries are located in <*DDS install dir*>/lib/<*architecture*>/., where <*DDS install dir*> is where you installed *RTI Data Distribution Service*, such as /local/rti/ndds.4.5x.

Table 4.9 **Additional Libraries for using RTI TCP Transport APIs on UNIX-based Systems with TLS Enabled**

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

a. The libraries are located in <TLS install dir>/lib/<architecture>/., where <TLS install dir> is where you installed RTI TLS Support, such as /local/rti/ndds.4.5x.

5 LynxOS Platforms

Table 5.1 on page 29 lists the architectures supported on LynxOS® operating systems.

Table 5.2 on page 29 and Table 5.3 on page 30 list the compiler flags and libraries you will need to link into your application.

Table 5.4 on page 31 provides details on the environment variables required to be set at run time for a LynxOS architecture.

Table 5.5 on page 31 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.

5.1 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 *RTI Data Distribution Service* 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 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.)

5.2 Supported Transports

Shared memory: Supported and enabled by default.

UDPv4: Supported and enabled by default.

UDPv6: Not supported.

TCP/IPv4: Not supported.

5.2.1 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 *RTI Data Distribution Service* are in the range of 0x400000. For example:

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

The shared semaphore keys used by *RTI Data Distribution Service* 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 *RTI Data Distribution Service*.

5.3 Serializable Support in Java

On LynxOS 4.0 systems, the default implementation of the Serializable Java interface does not work when the data types contain enumeration members.

5.4 Monotonic Clock Support

The monotonic clock (described in Section 8.6 in the *RTI Data Distribution Service User's Manual*) is not supported on LynxOS platforms.

5.5 Support for Controlling CPU Core Affinity for RTI Threads

Support for controlling CPU core affinity (described in Section 17.5 in the *RTI Data Distribution Service User's Manual*) is not available for LynxOS platforms.

Table 5.1 **Supported LynxOS Platforms**

Operating System	CPU	Compiler	RTI Architecture
LynxOS 4.0	Pentium class	gcc 3.2.2	i86Lynx4.0.0gcc3.2.2
		Sun Java Platform Standard Edition JDK 1.4	i86Lynx4.0.0gcc3.2.2jdk
	PPC 74xx (such as 7410)	gcc 3.2.2	ppc7400Lynx4.0.0gcc3.2.2
		Sun Java Platform Standard Edition JDK 1.4	ppc7400Lynx4.0.0gcc3.2.2jdk
	PPC 604 PPC 7XX (such as 750)	gcc 3.2.2	ppc750Lynx4.0.0gcc3.2.2
		Sun Java Platform Standard Edition JDK 1.4	ppc750Lynx4.0.0gcc3.2.2jdk
LynxOS 4.2	Pentium class	gcc 3.2.2	i86Lynx4.2.0gcc3.2.2
	PPC 74xx (such as 7410)	gcc 3.2.2	ppc7400Lynx4.2.0gcc3.2.2
LynxOS 5.0	PPC 74xx (such as 7410)	gcc 3.4.3	ppc7400Lynx5.0.0gcc3.4.3
		Sun Java Platform Standard Edition JDK 1.4	ppc7400Lynx5.0.0gcc3.4.3jdk
LynxOS-SE 3.0	Pentium class	gcc 3.4.3	i86LynxOS_SE3.0.0gcc3.4.3

Table 5.2 **Building Instructions for LynxOS Architectures (RTI Data Distribution Service Libraries/Jar Files)**

API	Library Format[a]	Required RTI Libraries or Jar Files[b]
C++	Static Release	libnddscppz.a libnddscz.a libnddscorez.a
	Static Debug	libnddscppzd.a libnddsczd.a libnddscorezd.a
	Dynamic Release	libnddscpp.so libnddsc.so libnddscore.so
	Dynamic Debug	libnddscppd.so libnddscd.so libnddscored.so

Table 5.2 **Building Instructions for LynxOS Architectures (RTI Data Distribution Service Libraries/Jar Files)**

API	Library Format[a]	Required RTI Libraries or Jar Files[b]
C	Static Release	libnddscz.a libnddscorez.a
	Static Debug	libnddsczd.a libnddscorezd.a
	Dynamic Release	libnddsc.so libnddscore.so
	Dynamic Debug	libnddscd.so libnddscored.so
Java	Release	nddsjava.jar
	Debug	nddsjavad.jar

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

b. The *RTI Data Distribution Service* C/C++ libraries are located in $(NDDSHOME)\lib\<*architecture*>\.
The *RTI Data Distribution Service* Java files are located in $(NDDSHOME)\class\.
(where $(NDDSHOME) is where *RTI Data Distribution Service* is installed, such as c:\rti\ndds.4.3x)

Table 5.3 **Building Instructions for LynxOS Architectures (System Libraries and Compiler Flags)**

API	RTI Architecture	Required System Libraries	Required Compiler Flags
C and C++	i86Lynx4.0.0gcc3.2.2	-ldb -lm -lrpc -lc -llynx	-DRTI_LYNX -mthreads -mshared
	i86Lynx4.2.0gcc3.2.2	-ldb -lm -lrpc -lc -llynx	
	i86LynxOS_SE3.0.0gcc3.4.3	-lm -lrpc -lc -lnetinet -lstdc++	-DRTI_LYNX -DRTI_LYNX_SE -mthreads -mshared
	ppc7400Lynx4.0.0gcc3.2.2	-ldb -lm -lrpc -lc -llynx	-DRTI_LYNX -mthreads -mshared
	ppc7400Lynx4.2.0gcc3.2.2		
	ppc7400Lynx5.0.0gcc3.4.3		
	ppc750Lynx4.0.0gcc3.2.2		
Java	i86Lynx4.0.0gcc3.2.2jdk	N/A	None
	ppc7400Lynx4.0.0gcc3.2.2jdk		
	ppc7400Lynx5.0.0gcc3.4.3jdk		
	ppc750Lynx4.0.0gcc3.2.2jdk		

Table 5.4 **Running Instructions for LynxOS Architectures**

RTI Architecture	Library Format	Required Environment Variables
All supported LynxOS architectures for Java:	N/A	LD_LIBRARY_PATH=${NDDSHOME}/lib/<*architecture*>:${LD_LIBRARY_PATH} [a]
All other supported LynxOS architectures	Static (Release and Debug)	None required
	Dynamic (Release and Debug)	LD_LIBRARY_PATH= ${NDDSHOME}/lib/<*architecture*>: ${LD_LIBRARY_PATH}[a]

a. ${NDDSHOME} represents the root directory of your *RTI Data Distribution Service* installation. ${LD_LIBRARY_PATH} represents the value of the LD_LIBRARY_PATH variable prior to changing it to support *RTI Data Distribution Service*. 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 5.5 **Library-Creation Details for LynxOS Architectures**

RTI Architecture	Library Format	Compiler Flags Used by RTI
i86Lynx4.0.0gcc3.2.2	Static and Dynamic Release	-mthreads -mshared -fPIC -D_POSIX_THREADS_CALLS -D__NO_INCLUDE_WARN__ -O -Wall -Wno-unknown-pragmas -DPtrIntType=long -DCPU=I80586 -DTARGET=\"i86Lynx4.0.0gcc3.2.2\" -DNDEBUG -c -Wp,-MD
	Static and Dynamic Debug	-mthreads -mshared -fPIC -D_POSIX_THREADS_CALLS -D__NO_INCLUDE_WARN__ -g -O -Wall -Wno-unknown-pragmas -DPtrIntType=long -DCPU=I80586 -DTARGET=\"i86Lynx4.0.0gcc3.2.2\" -c -Wp,-MD
i86Lynx4.2.0gcc3.2.2	Static and Dynamic Release	-mthreads -mshared -fPIC -D_POSIX_THREADS_CALLS -D__NO_INCLUDE_WARN__ -g -O -Wall -Wno-unknown-pragmas -DPtrIntType=long -DCSREAL_IS_FLOAT -DCPU=I80586 -DTARGET=\"i86Lynx4.2.0gcc3.2.2\" -DNDEBUG -c -Wp,-MD
	Static and Dynamic Debug	-mthreads -mshared -fPIC -D_POSIX_THREADS_CALLS -D__NO_INCLUDE_WARN__ -g -O -Wall -Wno-unknown-pragmas -DPtrIntType=long -DCSREAL_IS_FLOAT -DCPU=I80586 -DTARGET=\"i86Lynx4.2.0gcc3.2.2\" -c -Wp,-MD

Table 5.5 **Library-Creation Details for LynxOS Architectures**

RTI Architecture	Library Format	Compiler Flags Used by RTI
i86LynxOS_SE3.0.0gcc3.4.3	Static and Dynamic Release	-mthreads -mshared -fPIC -D_POSIX_THREADS_CALLS -D__NO_INCLUDE_WARN__ -O -Wall -Wno-unknown-pragmas -DPtrIntType=long -DCPU=I80586 -DTARGET=\"i86LynxOS_SE3.0.0gcc3.4.3\" -DNDEBUG -c -Wp,-MD
	Static and Dynamic Debug	-mthreads -mshared -fPIC -D_POSIX_THREADS_CALLS -D__NO_INCLUDE_WARN__ -g -O -Wall -Wno-unknown-pragmas -DPtrIntType=long -DCPU=I80586 -DTARGET=\"i86LynxOS_SE3.0.0gcc3.4.3\" -c -Wp,-MD
ppc7400Lynx4.0.0gcc3.2.2	Static and Dynamic Release	-mcpu=7400 -maltivec -mabi=altivec -fno-exceptions -mthreads -mshared -fPIC -D_POSIX_THREADS_CALLS -D__NO_INCLUDE_WARN__ -O -Wall -Wno-unknown-pragmas -DPtrIntType=long -DCPU=PPC7400 -DTARGET=\"ppc7400Lynx4.0.0gcc3.2.2\" -DNDEBUG -c -Wp,-MD
	Static and Dynamic Debug	-mcpu=7400 -maltivec -mabi=altivec -fno-exceptions -mthreads -mshared -fPIC -D_POSIX_THREADS_CALLS -D__NO_INCLUDE_WARN__ -g -O -Wall -Wno-unknown-pragmas -DPtrIntType=long -DCPU=PPC7400 -DTARGET=\"ppc7400Lynx4.0.0gcc3.2.2\" -c -Wp,-MD
ppc7400Lynx4.2.0gcc3.2.2	Static and Dynamic Release	-mcpu=7400 -maltivec -mabi=altivec -fno-exceptions -mthreads -mshared -fPIC -D_POSIX_THREADS_CALLS -D__NO_INCLUDE_WARN__ -O -Wall -Wno-unknown-pragmas -DPtrIntType=long -DCSREAL_IS_FLOAT -DCPU=PPC7400 -DTARGET=\"ppc7400Lynx4.2.0gcc3.2.2\" -DNDEBUG -c -Wp,-MD
	Static and Dynamic Debug	-mcpu=7400 -maltivec -mabi=altivec -fno-exceptions -mthreads -mshared -fPIC -D_POSIX_THREADS_CALLS -D__NO_INCLUDE_WARN__ -O -Wall -Wno-unknown-pragmas -DPtrIntType=long -DCSREAL_IS_FLOAT -DCPU=PPC7400 -DTARGET=\"ppc7400Lynx4.2.0gcc3.2.2\" -c -Wp,-MD

Table 5.5 **Library-Creation Details for LynxOS Architectures**

RTI Architecture	Library Format	Compiler Flags Used by RTI
ppc7400Lynx5.0.0gcc3.4.3	Static and Dynamic 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
	Static and Dynamic Debug	-mcpu=7400 -maltivec -mabi=altivec -fno-exceptions -mthreads -mshared -fPIC -D_POSIX_THREADS_CALLS -D__NO_INCLUDE_WARN__ -O -Wall -Wno-unknown-pragmas -DPtrIntType=long -DCSREAL_IS_FLOAT -DCPU=PPC7400 -DTARGET=\"ppc7400Lynx5.0.0gcc3.4.3\" -c -Wp,-MD
ppc750Lynx4.0.0gcc3.2.2	Static and Dynamic Release	-mcpu=750 -fno-exceptions -mthreads -mshared -fPIC -D_POSIX_THREADS_CALLS -D__NO_INCLUDE_WARN__ -O -Wall -Wno-unknown-pragmas -DPtrIntType=long -DCPU=PPC750 -DTARGET=\"ppc750Lynx4.0.0gcc3.2.2\" -DNDEBUG -c -Wp,-MD
	Static and Dynamic Debug	-mcpu=750 -fno-exceptions -mthreads -mshared -fPIC -D_POSIX_THREADS_CALLS -D__NO_INCLUDE_WARN__ -g -O -Wall -Wno-unknown-pragmas -DPtrIntType=long -DCPU=PPC750 -DTARGET=\"ppc750Lynx4.0.0gcc3.2.2\" -c -Wp,-MD
All supported LynxOS architectures for Java: i86Lynx4.0.0gcc3.2jdk ppc7400Lynx4.0.0gcc3.2.2jdk ppc7400Lynx5.0.0gcc3.4.3jdk ppc750Lynx4.0.0gcc3.2.2jdk	Dynamic Release	-target 1.4 –source 1.4
	Dynamic Debug	-target 1.4 –source 1.4 -g

6 Mac OS Platforms

Table 6.1 on page 35 lists the architectures supported on Mac OS operating systems.

Table 6.2 on page 35 lists the compiler flags and libraries you will need to link into your application.

Table 6.3 on page 36 provides details on the environment variables required to be set at run time for a Mac OS architecture.

Table 6.4 on page 36 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.

6.1 Multicast Support

Multicast is supported on Mac OS 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.

6.2 Supported Transports

Shared memory: Supported and enabled by default.

UDPv4: Supported and enabled by default.

UDPv6: Not supported.

TCP/IPv4: Not supported.

6.3 Monotonic Clock Support

The monotonic clock (described in Section 8.6 in the *RTI Data Distribution Service User's Manual*) is not supported on Mac OS platforms.

6.4 Support for Controlling CPU Core Affinity for RTI Threads

Support for controlling CPU core affinity (described in Section 17.5 in the *RTI Data Distribution Service User's Manual*) is not available for Mac OS platforms.

Table 6.1 **Mac OS Platforms**

Operating System	CPU	Compiler	RTI Architecture Abbreviation
Mac OS X	x64	gcc 4.2.1	x64Darwin10gcc4.2.1
		Java SE 1.6 for Mac OS	x64Darwin10gcc4.2.1jdk

Table 6.2 **Building Instructions for Mac OS Architectures**

API	Library Format	Required RTI Libraries[a]	Required System Libraries	Required Compiler Flags
C++	Static Release	libnddscppz.a libnddscz.a libnddscorez.a	-ldl -lm -lpthread	-dynamic -lpthread -lc -single_module
	Static Debug	libnddscppzd.a libnddsczd.a libnddscorezd.a		
	Dynamic Release	libnddscpp.dylib libnddsc.dylib libnddscore.dylib		
	Dynamic Debug	libnddscppd.dylib libnddscd.dylib libnddscored.dylib		
C	Static Release	libnddscz.a libnddscorez.a	-ldl -lm -lpthread	-dynamic -lpthread -lc -single_module
	Static Debug	libnddsczd.a libnddscorezd.a		
	Dynamic Release	libnddsc.dylib libnddscore.dylib		
	Dynamic Debug	libnddscd.dylib libnddscored.dylib		
Java	Release	nddsjava.jar	N/A	None required
	Debug	nddsjavad.jar		

a. The *RTI Data Distribution Service* C/C++ libraries are located in $(NDDSHOME)/lib/<architecture>/.
(where $(NDDSHOME) is where *RTI Data Distribution Service* is installed, such as /local/rti/ndds.4.5x)

Table 6.3 **Running Instructions for Mac OS Architectures**

RTI Architecture	Library Format (Release and Debug)	Required Environment Variables
x64Darwin10gcc4.2.1	Static	None required
	Dynamic	DYLD_LIBRARY_PATH=${NDDSHOME}/lib/<architecture>:${DYLD_LIBRARY_PATH}
x64Darwin10gcc4.2.1jdk	N/A	DYLD_LIBRARY_PATH=${NDDSHOME}/lib/<architecture>:${DYLD_LIBRARY_PATH}

Table 6.4 **Library-Creation Details for Mac OS Architectures**

RTI Architecture	Library Format (Static & Dynamic)	Compiler Flags Used by RTI
x64Darwin10gcc4.2.1	Release	-O -Wall -Wno-unknown-pragmas -DRTS_UNIX -DPtrIntType=long -DCSREAL_IS_FLOAT -DCPU=AMD64 -DTARGET=\"x64Darwin10gcc4.2.1\" -c -Wp,-MD
	Debug	-g -Wall -Wno-unknown-pragmas -DRTS_UNIX -DPtrIntType=long -DCSREAL_IS_FLOAT -DCPU=AMD64 -DTARGET=\" x64Darwin10gcc4.2.1\" -c -Wp,-MD
x64Darwin10gcc4.2.1jdk	Release	-target 1.6 -source 1.6
	Debug	-target 1.6 -source 1.6 -g

7 Solaris Platforms

Table 7.1 on page 39 lists the architectures supported on Solaris operating systems.

Table 7.2 on page 40 lists the compiler flags and the libraries you will need to link into your application. (See also: Libraries Required for using RTI Secure WAN Transport APIs (Section 7.5).)

Table 7.3 on page 41 provides details on the environment variables required to 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 7.4 on page 42 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.

7.1 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.

7.2 Supported Transports

Shared memory: Supported and enabled by default.

UDPv4: Supported and enabled by default.

UDPv6: Supported for all Solaris 2.9 and 2.10 platforms. The transport is not enabled by default, and the peers list must be modified to support IPv6. Note that mapping of the TransportPriority QoS is only supported for Solaris 2.10 platforms.

TCP/IPv4: Not supported.

7.2.1 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 *RTI Data Distribution Service* are in the range of 0x400000. For example:

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

The shared semaphore keys used by *RTI Data Distribution Service* 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 *RTI Data Distribution Service*.

7.2.2 Increasing Available Shared Resources

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

Each *RTI Data Distribution Service* 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 *RTI Data Distribution Service*, 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 *RTI Data Distribution Service* application that has shared memory enabled allocates 4 groups of 1 semaphore each (per domain). So setting the two values to be the same

number will work fine as far as *RTI Data Distribution Service* 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 *RTI Data Distribution Service* applications. You probably will not need to increase the shared memory parameters, since the default allows 100 shared memory areas, enough for 50 applications.

7.3 Monotonic Clock Support

The monotonic clock (described in Section 8.6 in the *RTI Data Distribution Service User's Manual*) is supported on all Solaris architectures.

7.4 Support for Controlling CPU Core Affinity for RTI Threads

Support for controlling CPU core affinity (described in Section 17.5 in the *RTI Data Distribution Service User's Manual*) is not available for Solaris platforms.

7.5 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 *RTI Data Distribution Service* 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 Release Notes Installation Guide* for details. To use the WAN or Secure Transport APIs, link against the additional libraries from Table 7.6 on page 45. (Select the files appropriate for your chosen library format.)

Table 7.1 **Supported Solaris Platforms**

Operating System	CPU	Compiler or Software Development Kit	RTI Architecture
Solaris 2.9	Pentium class	gcc 3.3.2	i86Sol2.9gcc3.3.2
		Sun Java Platform Standard Edition JDK 1.5 or 1.6	i86Sol2.9jdk
	UltraSPARC	CC 5.4 (Forte Dev 7, Sun One Studio 7)	sparcSol2.9cc5.4
		gcc 3.2	sparcSol2.9gcc3.2
		gcc 3.3	sparcSol2.9gcc3.3
		Sun Java Platform Standard Edition JDK 1.5 or 1.6	sparcSol2.9jdk

Table 7.1 **Supported Solaris Platforms**

Operating System	CPU	Compiler or Software Development Kit	RTI Architecture
Solaris 10	AMD64	gcc 3.4.3	x64Sol2.10gcc3.4.3
		Sun Java Platform Standard Edition JDK 1.5 or 1.6	x64Sol2.10jdk
	Pentium class	gcc 3.4.4	i86Sol2.10gcc3.4.4
		Sun Java Platform Standard Edition JDK 1.5 or 1.6	i86Sol2.10jdk
	UltraSPARC	gcc3.4.2	sparcSol2.10gcc3.4.2
		Sun Java Platform Standard Edition JDK 1.5 or 1.6	sparcSol2.10jdk
	UltraSPARC (with native 64-bit support)	cc 5.8	sparc64Sol2.10cc5.8
		gcc3.4.2	sparc64Sol2.10gcc3.4.2
		Sun Java Platform Standard Edition JDK 1.5 or 1.6	sparc64Sol2.10jdk

Table 7.2 **Building Instructions for Solaris Architectures**

API	Library Format	RTI Libraries or Jar Files[a]	Required System Libraries	Required Compiler Flags
C	Static Release	libnddscz.a libnddscorez.a	sparc64Sol2.10cc5.8, x64Sol2.10gcc3.4.3 and sparc64Sol2.10gcc3.4.2: -ldl -lnsl -lsocket -lgen -lposix4 -lpthread -lm -lc All other architectures: -ldl -lnsl -lgenIO -lsocket -lgen -lposix4 -lpthread -lm -lc	All architectures: -DRTI_UNIX sparc64Sol2.10cc5.8: also add: -xarch=v9 x64Sol2.10gcc3.4.3 and sparc64Sol2.10gcc3.4.2: also add: -m64 all other architectures: also add: -m32
	Static Debug	libnddsczd.a libnddscorezd.a		
	Dynamic Release	libnddsc.so libnddscore.so		
	Dynamic Debug	libnddscd.so libnddscored.so		

Table 7.2 **Building Instructions for Solaris Architectures**

API	Library Format	RTI Libraries or Jar Files[a]	Required System Libraries	Required Compiler Flags
C++	Static Release	libnddscppz.a libnddscz.a libnddscorez.a	sparc64Sol2.10cc5.8, x64Sol2.10gcc3.4.3 and sparc64Sol2.10gcc3.4.2: -ldl -lnsl -lsocket -lgen -lposix4 -lpthread -lm -lc All other architectures: -ldl -lnsl -lgenIO -lsocket -lgen -lposix4 -lpthread -lm -lc	All architectures: -DRTI_UNIX sparc64Sol2.10cc5.8: also add: -xarch=v9 x64Sol2.10gcc3.4.3 and sparc64Sol2.10gcc3.4.2: also add: -m64 all other architectures: also add: -m32
	Static Debug	libnddscppzd.a libnddsczd.a libnddscorezd.a		
	Dynamic Release	libnddscpp.so libnddsc.so libnddscore.so		
	Dynamic Debug	libnddscppd.so libnddscd.so libnddscored.so		
Java	Release	nddsjava.jar	N/A	None required
	Debug	nddsjavad.jar		

a. The RTI C/C++ libraries are located in $(NDDSHOME)\lib\<*architecture*>\.
The RTI Java files are located in $(NDDSHOME)\class\.
(where $(NDDSHOME) is where *RTI Data Distribution Service* is installed, such as /local/rti/ndds/ndds.4.5x)

Table 7.3 **Running Instructions for Solaris Architectures**

RTI Architecture	Library Format	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 (...64...jdk), use **-d64** in the command line.
All supported Solaris native architectures	Static (Release and Debug)	None required
	Dynamic (Release and Debug)	LD_LIBRARY_PATH= ${NDDSHOME}/lib/<*architecture*>: ${LD_LIBRARY_PATH} [a]

a. ${NDDSHOME} represents the root directory of your *RTI Data Distribution Service* installation.
${LD_LIBRARY_PATH} represents the value of the LD_LIBRARY_PATH variable prior to changing it to support *RTI Data Distribution Service*. 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 7.4 **Library-Creation Details for Solaris Architectures**

RTI Architecture	Library Format	Compiler Flags Used by RTI
i86Sol2.9gcc3.3.2[a]	Static and Dynamic Release	-D_POSIX_C_SOURCE=199506L -D__EXTENSIONS__ -DSolaris2 -DSVR5 -DSUN4_SOLARIS2 -O -Wall -Wno-unknown-pragmas -fPIC -DRTS_UNIX -DPtrIntType=long -DCSREAL_IS_FLOAT -DCPU=i386 -DTARGET=\"i86Sol2.9gcc3.3.2\" -DNDEBUG -c -Wp,-MD -Wp
	Static and Dynamic Debug	-D_POSIX_C_SOURCE=199506L -D__EXTENSIONS__ -DSolaris2 -DSVR5 -DSUN4_SOLARIS2 -g -O -Wall -Wno-unknown-pragmas -fPIC -DRTS_UNIX -DPtrIntType=long -DCSREAL_IS_FLOAT -DCPU=i386 -DTARGET=\"i86Sol2.9gcc3.3.2\" -c -Wp,-MD -Wp
i86Sol2.10gcc3.4.4[a]	Static and Dynamic Release	-D_POSIX_C_SOURCE=199506L -D__EXTENSIONS__ -DSolaris2 -DSVR5 -DSUN4_SOLARIS2 -O -Wall -Wno-unknown-pragmas -DRTS_UNIX -DPtrIntType=long -DCPU=I80586 -DTARGET=\"i86Sol2.10gcc3.4.4\" -DNDEBUG -c -Wp,-MD -Wp
	Static and Dynamic Debug	-D_POSIX_C_SOURCE=199506L -D__EXTENSIONS__ -DSolaris2 -DSVR5 -DSUN4_SOLARIS2 -g -O -Wall -Wno-unknown-pragmas -DRTS_UNIX -DPtrIntType=long -DCPU=I80586 -DTARGET=\"i86Sol2.10gcc3.4.4\" -c -Wp,-MD -Wp
sparcSol2.9cc5.4	Static and Dynamic Release	-D_POSIX_C_SOURCE=199506L -D__EXTENSIONS__ -DSolaris2 -DSVR5 -DSUN4_SOLARIS2 -KPIC -O +w -DRTS_UNIX -DPtrIntType=long -DCPU=SPARC -DTARGET=\"sparcSol2.9cc5.4\" -DNDEBUG -c
	Static and Dynamic Debug	-D_POSIX_C_SOURCE=199506L -D__EXTENSIONS__ -DSolaris2 -DSVR5 -DSUN4_SOLARIS2 -KPIC -g +w -DRTS_UNIX -DPtrIntType=long -DCPU=SPARC -DTARGET=\"sparcSol2.9cc5.4\" -c
sparcSol2.9gcc3.2[a]	Static and Dynamic Release	-D_POSIX_C_SOURCE=199506L -D__EXTENSIONS__ -DSolaris2 -DSVR5 -DSUN4_SOLARIS2 -O -Wall -Woverloaded-virtual -Wno-unknown-pragmas -DRTS_UNIX -DPtrIntType=long -DCPU=SPARC -DTARGET=\"sparcSol2.9gcc3.2\" -DNDEBUG -c -Wp, -MD
	Static and Dynamic Debug	-D_POSIX_C_SOURCE=199506L -D__EXTENSIONS__ -DSolaris2 -DSVR5 -DSUN4_SOLARIS2 -g -O -Wall -Woverloaded-virtual -Wno-unknown-pragmas -DRTS_UNIX -DPtrIntType=long -DCPU=SPARC -DTARGET=\"sparcSol2.9gcc3.2\" -c -Wp,-MD

Table 7.4 **Library-Creation Details for Solaris Architectures**

RTI Architecture	Library Format	Compiler Flags Used by RTI
sparcSol2.9gcc3.3[a]	Static and Dynamic Release	-D_POSIX_C_SOURCE=199506L -D__EXTENSIONS__ -DSolaris2 -DSVR5 -DSUN4_SOLARIS2 -O -Wall -Woverloaded-virtual -Wno-unknown-pragmas -DRTS_UNIX -DPtrIntType=long -DCPU=SPARC -DTARGET=\"sparcSol2.9gcc3.3\" -DNDEBUG -c -Wp, -MD
	Static and Dynamic Debug	-D_POSIX_C_SOURCE=199506L -D__EXTENSIONS__ -DSolaris2 -DSVR5 -DSUN4_SOLARIS2 -g -O -Wall -Woverloaded-virtual -Wno-unknown-pragmas -DRTS_UNIX -DPtrIntType=long -DCPU=SPARC -DTARGET=\"sparcSol2.9gcc3.3\" -c -Wp,-MD
sparcSol2.10gcc3.4.2[a]	Static and Dynamic Release	-D_POSIX_C_SOURCE=199506L -D__EXTENSIONS__ -DSolaris2 -DSVR5 -DSUN4_SOLARIS2 -O -Wall -Woverloaded-virtual -Wno-unknown-pragmas -DRTS_UNIX -DPtrIntType=long -DCSREAL_IS_FLOAT -DCPU=SPARC -DTARGET=\"sparcSol2.10gcc3.4.2\" -DNDEBUG -c -Wp, -MD
	Static and Dynamic Debug	-D_POSIX_C_SOURCE=199506L -D__EXTENSIONS__ -DSolaris2 -DSVR5 -DSUN4_SOLARIS2 -g -O -Wall -Woverloaded-virtual -Wno-unknown-pragmas -DRTS_UNIX -DPtrIntType=long -DCSREAL_IS_FLOAT -DCPU=SPARC -DTARGET=\"sparcSol2.10gcc3.4.2\" -c -Wp,-MD
sparc64Sol2.10cc5.8	Static and Dynamic Release	-xarch=v9 -fPIC -D_POSIX_C_SOURCE=199506L -D__EXTENSIONS__ -DSolaris2 -DSVR5 -DSUN4_SOLARIS2 -O -Wall -Wno-unknown-pragmas -DRTS_UNIX -DPtrIntType=long -DCSREAL_IS_FLOAT -DCPU=SPARC -DTARGET=\"sparc64Sol2.10cc5.8\" -DNDEBUG -c -Wp, -MD
	Static and Dynamic Debug	-xarch=v9 -fPIC -D_POSIX_C_SOURCE=199506L -D__EXTENSIONS__ -DSolaris2 -DSVR5 -DSUN4_SOLARIS2 -g -O -Wall -Wno-unknown-pragmas -DRTS_UNIX -DPtrIntType=long -DCSREAL_IS_FLOAT -DCPU=SPARC -DTARGET=\"sparc64Sol2.10cc5.8\" -c -Wp, -MD

Table 7.4 **Library-Creation Details for Solaris Architectures**

RTI Architecture	Library Format	Compiler Flags Used by RTI
sparc64Sol2.10gcc3.4.2[a]	Static and Dynamic Release	-m64 -fPIC -D_POSIX_C_SOURCE=199506L -D__EXTENSIONS__ -DSolaris2 -DSVR5 -DSUN4_SOLARIS2 -O -Wall -Wno-unknown-pragmas -DRTS_UNIX -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 -DRTS_UNIX -DPtrIntType=long -DCSREAL_IS_FLOAT -DCPU=SPARC -DTARGET=\"sparc64Sol2.10gcc3.4.2\" -c -Wp, -MD
x64Sol2.10gcc3.4.3 [a]	Static and Dynamic Release	-m64 -m128bit-long-double -fPIC -D_POSIX_C_SOURCE=199506L -D__EXTENSIONS__ -DSolaris2 -DSVR5 -DSUN4_SOLARIS2 -O -Wall -Wno-unknown-pragmas -DRTS_UNIX -DPtrIntType=long -DCSREAL_IS_FLOAT -DCPU=AMD64 -DNDEBUG -DTARGET=\"x64Sol2.10gcc3.4.3\" -c -Wp,-MD
	Static and Dynamic Debug	-m64 -m128bit-long-double -fPIC -D_POSIX_C_SOURCE=199506L -D__EXTENSIONS__ -DSolaris2 -DSVR5 -DSUN4_SOLARIS2 -g -O -Wall -Wno-unknown-pragmas -DRTS_UNIX -DPtrIntType=long -DCSREAL_IS_FLOAT -DCPU=AMD64 -DTARGET=\"x64Sol2.10gcc3.4.3\" -c -Wp,-MD
i86Sol2.9jdk i86Sol2.10jdk sparcSol2.9jdk sparcSol2.10jdk	Dynamic Release	-target 1.4 –source 1.4
	Dynamic Debug	-target 1.4 –source 1.4 -g
x64Sol2.10jdk sparc64Sol2.10jdk	Dynamic Release	-target 1.6 -source 1.6
	Dynamic Debug	-target 1.6 -source 1.6 -g

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

Table 7.6 **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.a libcrypto.a
Dynamic Debug	libnddstransportwand.so libnddstransporttlsd.so	
Static Release	libnddstransporttlsz.a libnddstransporttlszd.a	
Static Debug	libnddstransportwanz.a libnddstransportwanzd.a	

a. The libraries are located in <*wan install dir*>/lib/<*architecture*>/. (where <*wan install dir*> is where you installed RTI Secure WAN Transport, such as /local/rti/ndds.4.5x)

b. These libraries are located <*openssl install dir*>/<*architecture*/lib, where <*openssl install dir*> is where you installed OpenSSL, such as /local/rti/openssl-0.9.8f.

8 VxWorks Platforms

Table 8.1 on page 48 lists the architectures supported on VxWorks operating systems. You can build a VxWorks application by cross-compiling from your development host.

Table 8.2 on page 50 lists the libraries you will need to link into your application, and required compiler flags.

Compiling an *RTI Data Distribution Service* application for VxWorks depends on the development platform. For more information, such as specific compiler flags, see the *VxWorks Programmer's Guide*. Table 8.3 on page 51 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*.

8.1 Increasing the Stack Size

RTI Data Distribution Service applications may require more than the default stack size on VxWorks.

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

8.2 Libraries for RTP on VxWorks 6

When using VxWorks 6 with Real Time Processes (RTP mode), please note the following limitations:

- ❏ Dynamic libraries are not available for VxWorks 6.0, 6.1 and 6.2 systems with RTP mode (due to a bug in the gcc 3.3.2 tool chain).
- ❏ Dynamic libraries are not available for VxWorks 6.3 and higher systems with RTP mode on PPC CPUs.
- ❏ Dynamic libraries for RTP mode on VxWorks 6.3 and higher systems with Pentium CPUs *are* available.

8.3 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 *RTI Data Distribution Service* applications running on other systems that were talking to the restarted application. If all the *RTI Data Distribution Service* applications are restarted, there should be no problem.

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

All *RTI Data Distribution Service* 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.

8.4 Multicast Support

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

8.5 Supported Transports

Shared memory: Shared memory is supported and enabled by default on all VxWorks 6.x architectures.

UDPv4: Supported and enabled by default.

UDPv6: Not supported.

TCP/IPv4: Not supported.

8.6 Monotonic Clock Support

The monotonic clock (described in Section 8.6 in the *RTI Data Distribution Service User's Manual*) is supported on VxWorks 6.3 and higher architectures.

8.7 Support for Controlling CPU Core Affinity for RTI Threads

Support for controlling CPU core affinity (described in Section 17.5 in the *RTI Data Distribution Service User's Manual*) is not available for VxWorks platforms.

Table 8.1 **Supported VxWorks Target Platforms**[a]

Operating System	CPU	Compiler	RTI Architecture
VxWorks 5.4.2	ppc604, ppc750, ppc7400	gcc 2.96	ppc604Vx5.4gcc
VxWorks 5.5.1	Pentium	gcc 2.96	pentiumVx5.5gcc
	ppc405		ppc405Vx5.5gcc
	ppc604, ppc750, ppc7400		ppc604Vx5.5gcc
	ppc603		ppc603Vx5.5gcc
VxWorks 6.0, 6.1, 6.2	Pentium	gcc 3.3.2	For kernel modules: pentiumVx6.0gcc3.3.2 For Real Time Processes: pentiumVx6.0gcc3.3.2_rtp
	any Wind River PPC32 CPU with floating point hardware		For kernel modules: ppc604Vx6.0gcc3.3.2 For Real Time Processes: ppc604Vx6.0gcc3.3.2_rtp
VxWorks 6.3, 6.4	Pentium	gcc 3.4.4	For kernel modules: pentiumVx6.3gcc3.4.4 For Real Time Processes: pentiumVx6.3gcc3.4.4_rtp
	any Wind River PPC32 CPU with floating point hardware		For kernel modules: ppc604Vx6.3gcc3.4.4 For Real Time Processes: ppc604Vx6.3gcc3.4.4_rtp
VxWorks 6.5	Pentium	gcc 3.4.4	For kernel modules: pentiumVx6.5gcc3.4.4 For Real Time Processes: pentiumVx6.5gcc3.4.4_rtp
	any Wind River PPC32 CPU with floating point hardware		For kernel modules: ppc604Vx6.5gcc3.4.4 For Real Time Processes: ppc604Vx6.5gcc3.4.4_rtp

Table 8.1 **Supported VxWorks Target Platforms**[a]

Operating System	CPU	Compiler	RTI Architecture
VxWorks 6.6	Pentium	gcc 4.1.2	For Kernel Modules: pentiumVx6.6gcc4.1.2 For Real Time Processes: pentiumVx6.6gcc4.1.2_rtp
	any Wind River PPC32 CPU with floating point hardware	gcc 4.1.2	For Kernel Modules: ppc604Vx6.6gcc4.1.2 For Real Time Processes: ppc604Vx6.6gcc4.1.2_rtp
	ppc405[b]	gcc 4.1.2	For Kernel Modules: ppc405Vx6.6gcc4.1.2 For Real Time Processes: ppc405Vx6.6gcc4.1.2_rtp
VxWorks 6.7	Pentium	gcc 4.1.2	For Kernel Modules: pentiumVx6.7gcc4.1.2 For Real Time Processes: pentiumVx6.7gcc4.1.2_rtp
	any Wind River PPC32 CPU with floating point hardware	gcc 4.1.2	For Kernel Modules: ppc604Vx6.7gcc4.1.2 For Real Time Processes: ppc604Vx6.7gcc4.1.2_rtp
	ppc405[b]	gcc 4.1.2	For Kernel Modules: ppc405Vx6.6gcc4.1.2 For Real Time Processes: ppc405Vx6.6gcc4.1.2_rtp

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

b. For ppc405, the architecture string is the same for VxWorks 6.6 and 6.7.

Table 8.2 **Building Instructions for VxWorks Architectures**

API	Library Format	Required RTI Libraries[a]	Required Kernel Components	Required Compiler Flags
C++	Static Release	libnddscppz.a libnddscz.a libnddscorez.a	INCLUDE_TIMESTAMP For VxWorks 6.4 and below, also use: INCLUDE_ZBUF_SOCK INCLUDE_IGMP For VxWorks 6.3 and higher, also use: INCLUDE_POSIX_CLOCKS	-DRTI_VXWORKS
	Static Debug	libnddscppzd.a libnddsczd.a libnddscorezd.a		
	Dynamic Release	libnddscpp.so libnddsc.so libnddscore.so		
	Dynamic Debug	libnddscppd.so libnddscd.so libnddscored.so		
C	Static Release	libnddscz.a libnddscorez.a	INCLUDE_TIMESTAMP For VxWorks 6.4 and below, also use: INCLUDE_ZBUF_SOCK INCLUDE_IGMP For VxWorks 6.3 and higher, also use: INCLUDE_POSIX_CLOCKS	-DRTI_VXWORKS
	Static Debug	libnddsczd.a libnddscorezd.a		
	Dynamic Release	libnddsc.so libnddscore.so		
	Dynamic Debug	libnddscd.so libnddscored.so		

a. The *RTI Data Distribution Service* C/C++ libraries are located in $(NDDSHOME)\lib\<*architecture*>\.
(where $(NDDSHOME) is where *RTI Data Distribution Service* is installed, such as c:\rti\ndds.4.5x)

Table 8.3 **Library-Creation Details for VxWorks Architectures**

RTI Architecture	Library Format	Compiler Flags Used by RTI
pentiumVx5.5gcc	Static or Dynamic Release	-march=pentium -nostdlib -fno-defer-pop -fno-builtin -DTOOL_FAMILY=gnu -DTOOL=gnu -D__PROTOTYPE_5_0 -DVXWORKS_MAJOR_VERSION=5 -DVXWORKS_MINOR_VERSION=5 -O -Wall -Wno-unknown-pragmas -DRTS_VXWORKS -DPtrIntType=long -DCPU=PENTIUM -DNDEBUG -c -Wp,-MD
	Static or Dynamic Debug	-march=pentium -nostdlib -fno-defer-pop -fno-builtin -DTOOL_FAMILY=gnu -DTOOL=gnu -D__PROTOTYPE_5_0 -g -DVXWORKS_MAJOR_VERSION=5 -DVXWORKS_MINOR_VERSION=5 -O -Wall -Wno-unknown-pragmas -DRTS_VXWORKS -DPtrIntType=long -DCSREAL_IS_FLOAT -DCPU=PENTIUM -c -Wp,-MD
pentiumVx6.0gcc3.3.2	Static or Dynamic Release	-mcpu=pentium -fno-builtin -DTOOL=gnu -D_WRS_KERNEL -D__PROTOTYPE_5_0 -DVXWORKS_MAJOR_VERSION=6 -DVXWORKS_MINOR_VERSION=0 -O -Wall -Wno-unknown-pragmas -DRTS_VXWORKS -DPtrIntType=long -DCPU=PENTIUM -DNDEBUG -c -Wp,-MD
	Static or Dynamic Debug	-mcpu=pentium -fno-builtin -DTOOL=gnu -D_WRS_KERNEL -D__PROTOTYPE_5_0 -g -DVXWORKS_MAJOR_VERSION=6 -DVXWORKS_MINOR_VERSION=0 -O -Wall -Wno-unknown-pragmas -DRTS_VXWORKS -DPtrIntType=long -DCPU=PENTIUM -c -Wp,-MD
pentiumVx6.0gcc3.3.2_rtp	Static Release	-mcpu=pentium -fno-builtin -DTOOL=gnu -mrtp -D__PROTOTYPE_5_0 -DVXWORKS_MAJOR_VERSION=6 -DVXWORKS_MINOR_VERSION=0 -O -Wall -Wno-unknown-pragmas -DRTS_VXWORKS -DPtrIntType=long -DCPU=PENTIUM -DNDEBUG -c -Wp,-MD
	Static Debug	-mcpu=pentium -fno-builtin -DTOOL=gnu -mrtp -D__PROTOTYPE_5_0 -g -DVXWORKS_MAJOR_VERSION=6 -DVXWORKS_MINOR_VERSION=0 -O -Wall -Wno-unknown-pragmas -DRTS_VXWORKS -DPtrIntType=long -DCPU=PENTIUM -c -Wp,-MD
pentiumVx6.3gcc3.4.4	Static or Dynamic Release	-march=pentium -fno-builtin -ansi -DTOOL=gnu -D_WRS_KERNEL -D_PROTOTYPE_5_0 -DVXWORKS_MAJOR_VERSION=6 -DVXWORKS_MINOR_VERSION=3 -O -Wall -Wno-unknown-pragmas -DRTS_VXWORKS -DTOOL_FAMILY=gnu -DPtrIntType=long -DCPU=PENTIUM -DNDEBUG -c -Wp,-MD
	Static or Dynamic Debug	-march=pentium -fno-builtin -ansi -DTOOL=gnu -D_WRS_KERNEL -D_PROTOTYPE_5_0 -g -DVXWORKS_MAJOR_VERSION=6 -DVXWORKS_MINOR_VERSION=3 -O -Wall -Wno-unknown-pragmas -DRTS_VXWORKS -DTOOL_FAMILY=gnu -DPtrIntType=long -DCPU=PENTIUM -c -Wp,-MD

Table 8.3 **Library-Creation Details for VxWorks Architectures**

RTI Architecture	Library Format	Compiler Flags Used by RTI
pentiumVx6.3gcc3.4.4_rtp	Static Release	-march=i486 -ansi -DTOOL=gnu -mrtp -D_PROTOTYPE_5_0 -DVXWORKS_MAJOR_VERSION=6 -DVXWORKS_MINOR_VERSION=3 -O -Wall -Wno-unknown-pragmas -DRTS_VXWORKS -DTOOL_FAMILY=gnu -DPtrIntType=long -DCPU=PENTIUM -DNDEBUG -c -Wp,-MD
	Static Debug	-march=i486 -ansi -DTOOL=gnu -mrtp -D_PROTOTYPE_5_0 -g -DVXWORKS_MAJOR_VERSION=6 -DVXWORKS_MINOR_VERSION=3 -O -Wall -Wno-unknown-pragmas -DRTS_VXWORKS -DTOOL_FAMILY=gnu -DPtrIntType=long -DCPU=PENTIUM -c -Wp,-MD
	Dynamic Release	-march=i486 -ansi -DTOOL=gnu -mrtp -D_PROTOTYPE_5_0 -fPIC -DVXWORKS_MAJOR_VERSION=6 -DVXWORKS_MINOR_VERSION=3 -O -Wall -Wno-unknown-pragmas -DRTS_VXWORKS -DTOOL_FAMILY=gnu -DPtrIntType=long -DCPU=PENTIUM -DNDEBUG -c -Wp,-MD
	Dynamic Debug	-march=i486 -ansi -DTOOL=gnu -mrtp -fPIC -D_PROTOTYPE_5_0 -g -DVXWORKS_MAJOR_VERSION=6 -DVXWORKS_MINOR_VERSION=3 -O -Wall -Wno-unknown-pragmas -DRTS_VXWORKS -DTOOL_FAMILY=gnu -DPtrIntType=long -DCPU=PENTIUM -c -Wp,-MD
pentiumVx6.5gcc3.4.4	Static or Dynamic Release	-march=pentium -fno-builtin -ansi -DTOOL=gnu -D_WRS_KERNEL -D_PROTOTYPE_5_0 -DVXWORKS_MAJOR_VERSION=6 -DVXWORKS_MINOR_VERSION=5 -O -Wall -Wno-unknown-pragmas -DRTS_VXWORKS -DTOOL_FAMILY=gnu -DPtrIntType=long -DCPU=PENTIUM -DNDEBUG -c -Wp,-MD
	Static or Dynamic Debug	-march=pentium -fno-builtin -ansi -DTOOL=gnu -D_WRS_KERNEL -D_PROTOTYPE_5_0 -g -DVXWORKS_MAJOR_VERSION=6 -DVXWORKS_MINOR_VERSION=5 -O -Wall -Wno-unknown-pragmas -DRTS_VXWORKS -DTOOL_FAMILY=gnu -DPtrIntType=long -DCPU=PENTIUM -c -Wp,-MD

Table 8.3 **Library-Creation Details for VxWorks Architectures**

RTI Architecture	Library Format	Compiler Flags Used by RTI
pentiumVx6.5gcc3.4.4_rtp	Static Release	-march=i486 -ansi -DTOOL=gnu -mrtp -D_PROTOTYPE_5_0 -DVXWORKS_MAJOR_VERSION=6 -DVXWORKS_MINOR_VERSION=5 -O -Wall -Wno-unknown-pragmas -DRTS_VXWORKS -DTOOL_FAMILY=gnu -DPtrIntType=long -DCPU=PENTIUM -DNDEBUG -c -Wp,-MD
	Static Debug	-march=i486 -ansi -DTOOL=gnu -mrtp -D_PROTOTYPE_5_0 -g -DVXWORKS_MAJOR_VERSION=6 -DVXWORKS_MINOR_VERSION=5 -O -Wall -Wno-unknown-pragmas -DRTS_VXWORKS -DTOOL_FAMILY=gnu -DPtrIntType=long -DCPU=PENTIUM -c -Wp,-MD
	Dynamic Release	-march=i486 -ansi -DTOOL=gnu -mrtp -D_PROTOTYPE_5_0 -fPIC -DVXWORKS_MAJOR_VERSION=6 -DVXWORKS_MINOR_VERSION=5 -O -Wall -Wno-unknown-pragmas -DRTS_VXWORKS -DTOOL_FAMILY=gnu -DPtrIntType=long -DCPU=PENTIUM -DNDEBUG -c -Wp,-MD
	Dynamic Debug	-march=i486 -ansi -DTOOL=gnu -mrtp -D_PROTOTYPE_5_0 -fPIC -g -DVXWORKS_MAJOR_VERSION=6 -DVXWORKS_MINOR_VERSION=5 -O -Wall -Wno-unknown-pragmas -DRTS_VXWORKS -DTOOL_FAMILY=gnu -DPtrIntType=long -DCPU=PENTIUM -c -Wp,-MD
pentiumVx6.6gcc4.1.2	Static or Dynamic Release	-march=pentium -fno-builtin -ansi -DTOOL=gnu -D_WRS_KERNEL -D_PROTOTYPE_5_0 -DVXWORKS_MAJOR_VERSION=6 -DVXWORKS_MINOR_VERSION=6 -O -Wall -Wno-unknown-pragmas -DRTS_VXWORKS -DTOOL_FAMILY=gnu -DPtrIntType=long -DCPU=PENTIUM -DNDEBUG -c -Wp,-MD
	Static or Dynamic Debug	-march=pentium -fno-builtin -ansi -DTOOL=gnu -D_WRS_KERNEL -D_PROTOTYPE_5_0 -g -DVXWORKS_MAJOR_VERSION=6 -DVXWORKS_MINOR_VERSION=6 -O -Wall -Wno-unknown-pragmas -DRTS_VXWORKS -DTOOL_FAMILY=gnu -DPtrIntType=long -DCPU=PENTIUM -c -Wp,-MD

Table 8.3 **Library-Creation Details for VxWorks Architectures**

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

Table 8.3 **Library-Creation Details for VxWorks Architectures**

RTI Architecture	Library Format	Compiler Flags Used by RTI
pentiumVx6.7gcc4.1.2_rtp	Static Release	-march=i486 -ansi -DTOOL=gnu -mrtp -D_PROTOTYPE_5_0 -DVXWORKS_MAJOR_VERSION=6 -DVXWORKS_MINOR_VERSION=7 -O -Wall -Wno-unknown-pragmas -DRTS_VXWORKS -DTOOL_FAMILY=gnu -DPtrIntType=long -DCPU=PENTIUM -DNDEBUG -c -Wp,-MD
	Static Debug	-march=i486 -ansi -DTOOL=gnu -mrtp -D_PROTOTYPE_5_0 -g -DVXWORKS_MAJOR_VERSION=6 -DVXWORKS_MINOR_VERSION=7 -O -Wall -Wno-unknown-pragmas -DRTS_VXWORKS -DTOOL_FAMILY=gnu -DPtrIntType=long -DCPU=PENTIUM -c -Wp,-MD
	Dynamic Release	-march=i486 -ansi -DTOOL=gnu -mrtp -D_PROTOTYPE_5_0 -fPIC -DVXWORKS_MAJOR_VERSION=6 -DVXWORKS_MINOR_VERSION=7 -O -Wall -Wno-unknown-pragmas -DRTS_VXWORKS -DTOOL_FAMILY=gnu -DPtrIntType=long -DCPU=PENTIUM -DNDEBUG -c -Wp,-MD
	Dynamic Debug	-march=i486 -ansi -DTOOL=gnu -mrtp -D_PROTOTYPE_5_0 -fPIC -g -DVXWORKS_MAJOR_VERSION=6 -DVXWORKS_MINOR_VERSION=7 -O -Wall -Wno-unknown-pragmas -DRTS_VXWORKS -DTOOL_FAMILY=gnu -DPtrIntType=long -DCPU=PENTIUM -c -Wp,-MD
ppc405Vx5.5gcc	Static or Dynamic Release	-mcpu=405 -G 0 -fno-builtin -mlongcall -D__PROTOTYPE_5_0 -DVXWORKS_MAJOR_VERSION=5 -DVXWORKS_MINOR_VERSION=5 -O -Wall -Wno-unknown-pragmas -DRTS_VXWORKS -DPtrIntType=long -DCPU=PPC405 -DNDEBUG -c -Wp,-MD
	Static or Dynamic Debug	-mcpu=405 -G 0 -fno-builtin -mlongcall -D__PROTOTYPE_5_0 -g -DVXWORKS_MAJOR_VERSION=5 -DVXWORKS_MINOR_VERSION=5 -O -Wall -Wno-unknown-pragmas -DRTS_VXWORKS -DPtrIntType=long -DCPU=PPC405 -c -Wp,-MD

Table 8.3 **Library-Creation Details for VxWorks Architectures**

RTI Architecture	Library Format	Compiler Flags Used by RTI
ppc405Vx6.6gcc4.1.2	Static or Dynamic Release	-mcpu=405 -fno-builtin -mlongcall -DTOOL=gnu -mstrict-align -msoft-float -ansi -D_WRS_KERNEL -D_PROTOTYPE_5_0 -DVXWORKS_MAJOR_VERSION=6 -DVXWORKS_MINOR_VERSION=6 -O -Wall -Wno-unknown-pragmas -DRTS_VXWORKS -DTOOL=gnu -DTOOL_FAMILY=gnu -DPtrIntType=long -DCPU=PPC405 -DNDEBUG -c -Wp,-MD
	Static or Dynamic Debug	-mcpu=405 -fno-builtin -mlongcall -DTOOL=gnu -mstrict-align -msoft-float -ansi -D_WRS_KERNEL -D_PROTOTYPE_5_0 -g -DVXWORKS_MAJOR_VERSION=6 -DVXWORKS_MINOR_VERSION=6 -O -Wall -Wno-unknown-pragmas -DRTS_VXWORKS -DTOOL=gnu -DTOOL_FAMILY=gnu -DPtrIntType=long -DCPU=PPC405 -c -Wp,-MD
ppc405Vx6.6gcc4.1.2_rtp	Static Release	-msoft-float -mlongcall -mregnames -mstrict-align -ansi -DTOOL=gnu -mrtp -D_PROTOTYPE_5_0 -DVXWORKS_MAJOR_VERSION=6 -DVXWORKS_MINOR_VERSION=6 -O -Wall -Wno-unknown-pragmas -DRTS_VXWORKS -DTOOL=sfgnu -DTOOL_FAMILY=gnu -DPtrIntType=long -DCPU=PPC32 -DNDEBUG -c -Wp,-MD
	Static Debug	-msoft-float -mlongcall -mregnames -mstrict-align -ansi -DTOOL=gnu -mrtp -fPIC -shared -D_PROTOTYPE_5_0 -g -DVXWORKS_MAJOR_VERSION=6 -DVXWORKS_MINOR_VERSION=6 -O -Wall -Wno-unknown-pragmas -DRTS_VXWORKS -DTOOL=sfgnu -DTOOL_FAMILY=gnu -DPtrIntType=long -DCPU=PPC32 -c -Wp,-MD
ppc603Vx5.5gcc	Static or Dynamic Release	-mcpu=603 -G 0 -fno-builtin -mlongcall -D__PROTOTYPE_5_0 -DVXWORKS_MAJOR_VERSION=5 -DVXWORKS_MINOR_VERSION=5 -O -Wall -Wno-unknown-pragmas -DRTS_VXWORKS -DPtrIntType=long -DCSREAL_IS_FLOAT -DCPU=PPC603 -DNDEBUG -c -Wp,-MD
	Static or Dynamic Debug	-mcpu=603 -G 0 -fno-builtin -mlongcall -D__PROTOTYPE_5_0 -g -DVXWORKS_MAJOR_VERSION=5 -DVXWORKS_MINOR_VERSION=5 -O -Wall -Wno-unknown-pragmas -DRTS_VXWORKS -DPtrIntType=long -DCSREAL_IS_FLOAT -DCPU=PPC603 -c -Wp,-MD
ppc604Vx5.4gcc	Static or Dynamic Release	-mcpu=604 -mstrict-align -fstrength-reduce -fno-builtin -mlongcall -nostdinc -D__PROTOTYPE_5_0 -DVXWORKS_MAJOR_VERSION=5 -DVXWORKS_MINOR_VERSION=4 -O -Wall -Wno-unknown-pragmas -DRTS_VXWORKS -DPtrIntType=long -DCPU=PPC604 -DNDEBUG -c -Wp,-MD
	Static or Dynamic Debug	-mcpu=604 -mstrict-align -fstrength-reduce -fno-builtin -mlongcall -nostdinc -D__PROTOTYPE_5_0 -g -DVXWORKS_MAJOR_VERSION=5 -DVXWORKS_MINOR_VERSION=4 -O -Wall -Wno-unknown-pragmas -DRTS_VXWORKS -DPtrIntType=long -DCPU=PPC604 -c -Wp,-MD

Table 8.3 **Library-Creation Details for VxWorks Architectures**

RTI Architecture	Library Format	Compiler Flags Used by RTI
ppc604Vx5.5gcc	Static or Dynamic Release	-mcpu=604 -G 0 -fno-builtin -mlongcall -D__PROTOTYPE_5_0 -DVXWORKS_MAJOR_VERSION=5 -DVXWORKS_MINOR_VERSION=5 -O -Wall -Wno-unknown-pragmas -DRTS_VXWORKS -DPtrIntType=long -DCPU=PPC604 -DNDEBUG -c -Wp,-MD
	Static or Dynamic Debug	-mcpu=604 -G 0 -fno-builtin -mlongcall -D__PROTOTYPE_5_0 -g -DVXWORKS_MAJOR_VERSION=5 -DVXWORKS_MINOR_VERSION=5 -O -Wall -Wno-unknown-pragmas -DRTS_VXWORKS -DPtrIntType=long -DCPU=PPC604 -c -Wp,-MD
ppc604Vx6.0gcc3.3.2	Static or Dynamic Release	-mcpu=604 -fno-builtin -mlongcall -DTOOL=gnu -D_WRS_KERNEL -D__PROTOTYPE_5_0 -DVXWORKS_MAJOR_VERSION=6 -DVXWORKS_MINOR_VERSION=0 -O -Wall -Wno-unknown-pragmas -DRTS_VXWORKS -DPtrIntType=long -DCPU=PPC604 -DNDEBUG -c -Wp,-MD
	Static or Dynamic Debug	-mcpu=604 -fno-builtin -mlongcall -DTOOL=gnu -D_WRS_KERNEL -D__PROTOTYPE_5_0 -g -DVXWORKS_MAJOR_VERSION=6 -DVXWORKS_MINOR_VERSION=0 -O -Wall -Wno-unknown-pragmas -DRTS_VXWORKS -DPtrIntType=long -DCPU=PPC604 -c -Wp,-MD
ppc604Vx6.0gcc3.3.2_rtp	Static Release	-mcpu=604 -fno-builtin -mlongcall -DTOOL=gnu -mrtp -D__PROTOTYPE_5_0 -DVXWORKS_MAJOR_VERSION=6 -DVXWORKS_MINOR_VERSION=0 -O -Wall -Wno-unknown-pragmas -DRTS_VXWORKS -DPtrIntType=long -DCPU=PPC604 -DNDEBUG -c -Wp,-MD
	Static Debug	-mcpu=604 -fno-builtin -mlongcall -DTOOL=gnu -mrtp -D__PROTOTYPE_5_0 -g -DVXWORKS_MAJOR_VERSION=6 -DVXWORKS_MINOR_VERSION=0 -O -Wall -Wno-unknown-pragmas -DRTS_VXWORKS -DPtrIntType=long -DCPU=PPC604 -c -Wp,-MD
ppc604Vx6.3gcc3.4.4	Static or Dynamic Release	-mcpu=604 -fno-builtin -mlongcall -DTOOL=gnu -mstrict-align -mno-implicit-fp -ansi -D_WRS_KERNEL -D_PROTOTYPE_5_0 -DVXWORKS_MAJOR_VERSION=6 -DVXWORKS_MINOR_VERSION=3 -O -Wall -Wno-unknown-pragmas -DRTS_VXWORKS -DTOOL_FAMILY=gnu -DPtrIntType=long -DCPU=PPC604 -DNDEBUG -c -Wp,-MD
	Static or Dynamic Debug	-mcpu=604 -fno-builtin -mlongcall -DTOOL=gnu -mstrict-align -mno-implicit-fp -ansi -D_WRS_KERNEL -D_PROTOTYPE_5_0 -g -DVXWORKS_MAJOR_VERSION=6 -DVXWORKS_MINOR_VERSION=3 -O -Wall -Wno-unknown-pragmas -DRTS_VXWORKS -DTOOL_FAMILY=gnu -DPtrIntType=long -DCPU=PPC604 -c -Wp,-MD

Table 8.3 **Library-Creation Details for VxWorks Architectures**

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

Table 8.3 **Library-Creation Details for VxWorks Architectures**

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

Table 8.3 **Library-Creation Details for VxWorks Architectures**

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

9 Windows Platforms

First, see the basic instructions for compiling on Microsoft® Windows® systems provided in Section 9.4 in the *RTI Data Distribution Service User's Manual*. The following tables provide supplemental information. Table 9.1 lists the architectures supported on Windows operating systems.

Table 9.1 **Supported Windows Architectures**

Operating System	CPU	Compiler or Software Development Kit [a] [b]	RTI Architecture
Windows 7	x86	Visual Studio 2010	i86Win32VS2010
		Sun Java Platform Standard Edition JDK 1.5 or 1.6	i86Win32jdk
		Visual Studio 2005 (C++, C# 8.0 or 9.0) SP 1	i86Win32dotnet2.0
Windows 7 x64 Edition	x64	Visual Studio 2010	x64Win64VS2010
		Sun Java Platform Standard Edition JDK 1.5 or 1.6	x64Win64jdk
		Visual Studio 2005 (C++, C# 8.0 or 9.0) SP 1	x64Win64dotnet2.0
Windows 2000	x86	Visual C 6.0	i86Win32VC60
		Visual C 7.0	i86Win32VC70
		Visual Studio 2003	i86Win32VS2003
		Visual Studio 2005 SP1	i86Win32VS2005
		Sun Java Platform Standard Edition JDK 1.5 or 1.6	i86Win32jdk
		Visual Studio 2005 (C++, C# 8.0 or 9.0) SP 1	i86Win32dotnet2.0
Windows 2003	x86	Visual C 6.0	i86Win32VC60
		Visual C 7.0	i86Win32VC70
		Visual Studio 2003	i86Win32VS2003
		Visual Studio 2008 SP1	i86Win32VS2008
		Sun Java Platform Standard Edition JDK 1.5 or 1.6	i86Win32jdk
		Visual Studio 2005 (C++, C# 8.0 or 9.0) SP 1	i86Win32dotnet2.0
Windows 2003 x64 Edition	x64	Visual Studio 2005 SP1	x64Win64VS2005
		Visual Studio 2008 SP1	x64Win64VS2008
		Sun Java Platform Standard Edition JDK 1.5 or 1.6	x64Win64jdk
		Visual Studio 2005 (C++, C# 8.0 or 9.0) SP 1	x64Win64dotnet2.0

Table 9.1 **Supported Windows Architectures**

Operating System	CPU	Compiler or Software Development Kit [a] [b]	RTI Architecture
Windows CE 6.0 (target only)[c] [d] [e]	armv4	Visual Studio 2005 (C++ 8.0) (Service Pack 1)	armv4WinCE6.0 VS2005
	x86		i86WinCE6.0VS2005
Windows Server 2008 R2, x64 Edition	x64	Visual Studio 2010	x64Win64VS2010
		Sun Java Platform Standard Edition JDK 1.5 or 1.6	x64Win64jdk
		Visual Studio 2005 (C++, C# 8.0 or 9.0) SP 1	x64Win64dotnet2.0
Windows Vista	x86	Visual Studio 2005 SP1	i86Win32VS2005
		Visual Studio 2008 SP1	i86Win32VS2008
		Sun Java Platform Standard Edition JDK 1.5 or 1.6	i86Win32jdk
		Visual Studio 2005 (C++, C# 8.0 or 9.0) SP 1	i86Win32dotnet2.0
Windows Vista 64-bit Edition	x64	Visual Studio 2005 SP1	x64Win64VS2005
		Visual Studio 2008 SP1	x64Win64VS2008
		Visual Studio 2005 (C++, C# 8.0 or 9.0) SP 1	x64Win64dotnet2.0
		Sun Java Platform Standard Edition JDK 1.5 or 1.6	x64Win64jdk
Windows XP Professional [f]	x86	Visual C 6.0	i86Win32VC60
		Visual C 7.0	i86Win32VC70
		Visual Studio 2003	i86Win32VS2003
		Visual Studio 2005 SP1	i86Win32VS2005
		Visual Studio 2008 SP1	i86Win32VS2008
		Sun Java Platform Standard Edition JDK 1.5 or 1.6	i86Win32jdk
		Visual Studio 2005 (C++, C# 8.0 or 9.0) SP 1	i86Win32dotnet2.0
Windows XP Professional x64 Edition	x64	Visual Studio 2005 SP1	x64Win64VS2005
		Visual Studio 2008 SP1	x64Win64VS2008
		Sun Java Platform Standard Edition JDK 1.5 or 1.6	x64Win64jdk
		Visual Studio 2005 (C++, C# 8.0 or 9.0) SP 1	x64Win64dotnet2.0

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

b. 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#.

c. The Windows CE network stack does not support the IP_TOS socket option.

d. The Windows CE device must be connected directly to the network, not through a Windows PC using ActiveSync. RTI Data Distribution Service does not support Windows CE when used with ActiveSync

e. On Windows CE systems, add registry values under HKEY_LOCAL_MACHINE\Software\RTI for the environment variables that RTI Data Distribution Service looks up. For example: To set NDDS_DISCOVERY_PEERS to 127.0.0.1, under "HKEY_LOCAL_MACHINE\Software\RTI", add a string value with the name "NDDS_DISCOVERY_PEERS" and data "127.0.0.1".

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

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

- ❏ Windows 2000, Windows 2003, Windows Vista, Windows XP (non-64-bit), Windows 7, Windows Server 2008 R2: Table 9.2 on page 68. (See also: Libraries Required for Using RTI Secure WAN Transport APIs (Section 9.8).)
- ❏ Windows XP Professional x64 Edition: Table 9.3 on page 69
- ❏ Windows CE: Table 9.4 on page 70

To use libraries that are *statically* linked into an application, link in all of the libraries listed in one of the rows of these tables. To use *dynamic* link libraries (DLL) on Windows systems, link in all of the libraries listed in one of the 'Dynamic' sections of the appropriate table. When the application executes, it will attempt to dynamically link in the libraries, which are located 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.

Visual Studio® 2005 — Service Pack 1 Requirement

- ❏ You must have Visual Studio 2005 Service Pack 1 or the Microsoft Visual C++ 2005 SP1 Redistribution Package installed on the machine where you are *running* an application built with the *release* libraries of the following RTI architecture packages:
  - x64Win64VS2005 built with dynamic libraries
  - i86Win32VS2005 built with dynamic libraries
  - i86Win32jdk, x64Win64jdk, i86Win32dotnet2.0, x64Win64dotnet2.0

If you are running an application built with the *debug* libraries of the above RTI architecture packages, you must have Visual Studio 2005 Service Pack 1 installed.

The Microsoft Visual C++ 2005 SP1 Redistribution Package can be downloaded from RTI Customer Portal, accessible from www.rti.com/support[1], or obtained from the following Microsoft website:

- For x86 architectures: http://www.microsoft.com/downloads/details.aspx?familyid=200B2FD9-AE1A-4A14-984D-389C36F85647&displaylang=en
- For x64 architectures: http://www.microsoft.com/downloads/details.aspx?familyID=EB4EBE2D-33C0-4A47-9DD4-B9A6D7BD44DA&displaylang=en

Visual Studio® 2008 - Service Pack 1 Requirement

❑ You must have Visual Studio 2008 Service Pack 1 or the Microsoft Visual C++ 2008 SP1 Redistribution Package installed on the machine where you are *running* an application built with the following RTI architecture packages:

- x64Win64VS2008 built with dynamic libraries
- i86Win32VS2008 built with dynamic libraries

The Microsoft Visual C++ 2008 SP1 Redistribution Package can be downloaded from RTI Customer Portal, accessible from www.rti.com/support[1], or obtained from the following Microsoft website:

- For x86 architectures: http://www.microsoft.com/downloads/details.aspx?familyid=A5C84275-3B97-4AB7-A40D-3802B2AF5FC2&displaylang=en
- For x64 architectures: http://www.microsoft.com/downloads/details.aspx?FamilyID=ba9257ca-337f-4b40-8c14-157cfdffee4e&displaylang=en

Windows Registry Setting for Better Performance

❑ On Windows systems, the following registry setting change will improve performance when sending UDP datagrams of size larger than 1024 bytes:

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

DWORD: Name=FastSendDatagramThreshold, Value = 65536

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

1. On the portal, select the Downloads page. The Redistribution Package is in the section labeled, "Windows Target Libraries for RTI Data Distribution Service."

Table 9.5 on page 71 provides details on the environment variables required to be set at run time for a Windows architecture.

When using dynamic libraries for Windows CE platforms: the required DLLs need to be present in the same folder as the application or in the '**\Windows**' folder.

For details on how the libraries were built by RTI, see Table 9.6 on page 72 (for Windows 2000 and Windows XP) and Table 9.7 on page 77 (for Windows CE). 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 9.8 on page 78 and Table 9.9 on page 78 list additional libraries required when using the optional *RTI Secure WAN Transport* and *RTI TCP Transport*, respectively.

9.1 Visual Studio 2005 Required when Using RTI 'Debug' Libraries for Java or .NET

The *RTI Data Distribution Service* dynamic libraries for Java and .NET rely on Microsoft Visual Studio 2005 Service Pack 1 run-time libraries. These libraries are available with Microsoft Visual Studio 2005 or as part of a redistributable package independent of Visual Studio. (The redistributable package is available for download from the RTI Customer Portal.)

However, while Microsoft includes debug versions of these run-time libraries with Visual Studio, it only includes *release* versions in the redistributable package. This limitation means that if you do not have Visual Studio 2005 installed, you cannot use the RTI *debug* libraries; you must use the RTI *release* libraries. If you attempt to use the RTI debug libraries, and your system does not have debug versions of the Microsoft run-time libraries available, your application will fail to start up properly. If you start it from a command shell, you will see an error about a failure to load the dynamic libraries.

Fortunately, you do not need to use the RTI debug libraries to debug your own code. If you experience library-loading problems when your Java or .NET application starts up in debug mode, modify your application project files to use the release versions of the RTI libraries. Alternatively, you can obtain a no-cost version of Visual Studio 2005 directly from Microsoft, which will contain the necessary debug libraries.

9.2 .NET API Requires Thread Affinity

To maintain proper concurrency control, .NET threads that call an *RTI Data Distribution Service* 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 *RTI Data Distribution Service* 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, you should call **Thread.EndThreadAffinity()**.

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

9.3 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.

Note: Windows CE 6.0 does not support multicast loopback.

9.4 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 2003 or higher. The peers list (**NDDS_DISCOVERY_PEERS**) must be modified to support UDPv6.

TCP/IPv4: Supported on architectures that use Visual Studio 2005 or higher. (This is *not* a built-in transport.)

9.5 Monotonic Clock Support

The monotonic clock (described in Section 8.6 in the *RTI Data Distribution Service User's Manual*) is supported.

9.6 Support for Controlling CPU Core Affinity for RTI Threads

Support for controlling CPU core affinity (described in Section 17.5 in the *RTI Data Distribution Service User's Manual*) is not available for Windows platforms.

9.7 PPP Link Support for Windows XP Systems

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

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

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

Failure to do so will result in *RTI Data Distribution Service* being unable to send any data at all over the PPP link.

Notes:

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

9.8 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 *RTI Data Distribution Service* 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 WAN or Secure Transport APIs, add the libraries from Table 9.8 on page 78 to your project files.

9.9 Libraries Required for Using RTI TCP Transport APIs

To use the TCP Transport APIs, link against the additional libraries from Table 9.9 on page 78. (Select the files appropriate for your chosen library format.)

Table 9.2 **Building Instructions for Windows Host Architectures**

API	Library Format	RTI Libraries or Jar Files[a]	Required System Libraries	Required Compiler Flags
C	Static Release	nddscz.lib nddscorez.lib	netapi32.lib advapi32.lib user32.lib ws2_32.lib iphlpapi.lib	-DRTI_WIN32 /MT
	Static Debug	nddsczd.lib nddscorezd.lib		-DRTI_WIN32 /MTd
	Dynamic Release	nddsc.lib nddscore.lib		-DRTI_WIN32 -DNDDS_DLL_VARIABLE /MD
	Dynamic Debug	nddscd.lib nddscored.lib		-DRTI_WIN32 -DNDDS_DLL_VARIABLE /MDd
C++	Static Release	nddscppz.lib nddscz.lib nddscorez.lib	netapi32.lib advapi32.lib user32.lib ws2_32.lib iphlpapi.lib	-DRTI_WIN32 /MT
	Static Debug	nddscppzd.lib nddsczd.lib nddscorezd.lib		-DRTI_WIN32 /MTd
	Dynamic Release	nddscpp.lib nddsc.lib nddscore.lib		-DRTI_WIN32 -DNDDS_DLL_VARIABLE /MD
	Dynamic Debug	nddscppd.lib nddscd.lib nddscored.lib		-DRTI_WIN32 -DNDDS_DLL_VARIABLE /MDd
C++/CLI	Release	nddscpp.lib nddsc.lib nddscore.lib nddsdotnet.dll	N/A	-DRTI_WIN32 -DNDDS_DLL_VARIABLE /MD -DWIN32_LEAN_AND_MEAN
	Debug	nddscppd.lib nddscd.lib nddscored.lib nddsdotnetd.dll		-DRTI_WIN32 -DNDDS_DLL_VARIABLE /MDd -DWIN32_LEAN_AND_MEAN
C#	Release	N/A	N/A	N/A
	Debug			

Table 9.2 **Building Instructions for Windows Host Architectures**

API	Library Format	RTI Libraries or Jar Files[a]	Required System Libraries	Required Compiler Flags
Java	Release	nddsjava.jar	N/A	N/A
	Debug	nddsjavad.jar		

a. The RTI C/C++ libraries are located in $(NDDSHOME)\lib\<*architecture*>\.
The RTI Java libraries are located in $(NDDSHOME)\class\.
(where $(NDDSHOME) is where *RTI Data Distribution Service* is installed, such as c:\rti\ndds.4.5x)

Table 9.3 **Building Instructions for Windows Target Architectures (Windows XP x64 Edition)**

API	Library Format	RTI Libraries or Jar Files[a]	Required System Libraries	Required Compiler Flags
C	Static Release	nddscz.lib nddscorez.lib	netapi32.lib advapi32.lib user32.lib ws2_32.lib iphlpapi.lib	/Gd /MT /D "WIN32" /D "RTI_WIN32" /D "NDEBUG"
	Static Debug	nddsczd.lib nddscorezd.lib		/Gd /MTd /D "WIN32" /D "RTI_WIN32"
	Dynamic Release	nddsc.lib nddscore.lib		/Gd /MD /D "WIN32" /D "NDDS_DLL_VARIABLE" /D "RTI_WIN32" /D "NDEBUG"
	Dynamic Debug	nddscd.lib nddscored.lib		/Gd /MDd /D "WIN32" /D "NDDS_DLL_VARIABLE" /D "RTI_WIN32"
C++	Static Release	nddscppz.lib nddscz.lib nddscorez.lib	netapi32.lib advapi32.lib user32.lib ws2_32.lib iphlpapi.lib	/Gd /EHsc /MT /D "WIN32" /D "RTI_WIN32" /D "NDEBUG"
	Static Debug	nddscppzd.lib nddsczd.lib nddscorezd.lib		/Gd /EHsc /MTd /D "WIN32"/D "RTI_WIN32"
	Dynamic Release	nddscpp.lib nddsc.lib nddscore.lib		/Gd /EHsc /MD /D "WIN32" /D "NDDS_DLL_VARIABLE" /D "RTI_WIN32" /D "NDEBUG"
	Dynamic Debug	nddscppd.lib nddscd.lib nddscored.lib		/Gd /EHsc /MDd /D "WIN32" /D "NDDS_DLL_VARIABLE" /D "RTI_WIN32"

Table 9.3 **Building Instructions for Windows Target Architectures (Windows XP x64 Edition)**

API	Library Format	RTI Libraries or Jar Files[a]	Required System Libraries	Required Compiler Flags
C++/CLI, C#	Release	nddscpp.lib nddsc.lib nddscore.lib	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		/Gd /EHsc /MDd /D "WIN32" /D "NDDS_DLL_VARIABLE" /D "RTI_WIN32"
Java	Release	nddsjava.jar	N/A	N/A
	Debug	nddsjavad.jar		

a. The RTI C/C++ libraries are located in $(NDDSHOME)\lib\<*architecture*>\.
The RTI Java libraries are located in $(NDDSHOME)\class\.
(where $(NDDSHOME) is where *RTI Data Distribution Service* is installed, such as c:\rti\ndds.4.5x

Table 9.4 **Building Instructions for Windows Target Architectures (Windows CE)**

API	Library Format	RTI Libraries or Jar Files[a]	Required System Libraries	Required Compiler Flags
C	Static Release	nddscz.lib nddscorez.lib	coredll.lib corelibc.lib ws2.lib iphlpapi.lib	-DRTI_WINCE /MT
	Static Debug	nddsczd.lib nddscorezd.lib		-DRTI_WINCE /MTd
	Dynamic Release	nddsc.lib nddscore.lib		-DRTI_WINCE -DNDDS_DLL_VARIABLE /MD
	Dynamic Debug	nddscd.lib nddscored.lib		-DRTI_WINCE -DNDDS_DLL_VARIABLE /MDd

Table 9.4 **Building Instructions for Windows Target Architectures (Windows CE)**

API	Library Format	RTI Libraries or Jar Files[a]	Required System Libraries	Required Compiler Flags
C++	Static Release	nddscppz.lib nddscz.lib nddscorez.lib	coredll.lib corelibc.lib ws2.lib iphlpapi.lib	-DRTI_WINCE /MT
	Static Debug	nddscppzd.lib nddsczd.lib nddscorezd.lib		-DRTI_WINCE /MTd
	Dynamic Release	nddscpp.lib nddsc.lib nddscore.lib		-DRTI_WINCE -DNDDS_DLL_VARIABLE /MD
	Dynamic Debug	nddscppd.lib nddscd.lib nddscored.lib		-DRTI_WINCE -DNDDS_DLL_VARIABLE /MDd

a. The RTI C/C++ libraries are located in $(NDDSHOME)\lib\<*architecture*>\.
(where $(NDDSHOME) is where *RTI Data Distribution Service* is installed, such as c:\rti\ndds.4.5x)

Table 9.5 **Running Instructions for Windows Architectures**

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

a. %NDDSHOME% represents the root directory of your *RTI Data Distribution Service* installation. %Path% represents the value of the Path variable prior to changing it to support *RTI Data Distribution Service*. When using nddsjava.jar, the Java virtual machine (JVM) will attempt to load release versions of the native libraries. When using nddsjavad.jar, the JVM will attempt to load debug versions of the native libraries.

Table 9.6 **Library-Creation Details for Windows Host Architectures**

RTI Architecture	Library Format	Compiler Flags Used by RTI
i86Win32dotnet2.0	Dynamic Release	/O2 /GL /D "WIN32" /D "NDEBUG" /D "RTI_WIN32" /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 "RTI_WIN32" /D "NDDS_DLL_VARIABLE" /D "_WINDLL" /D "_UNICODE" /D "UNICODE" /FD /EHa /MDd /c /Zi /clr /TP
i86Win32jdk	Dynamic Release	-target 1.4 –source 1.4
	Dynamic Debug	-target 1.4 –source 1.4 -g
i86Win32VC60	Static Release	-DPtrIntType=long -DCPU=I80586 -DTARGET=\"i86Win32VC60\" -DWIN32 -D_WINDOWS -D_WIN32_WINNT=0x0400 -DWIN32_LEAN_AND_MEAN /O2 /Zi /GX /MT -DNDEBUG -c
	Dynamic Release	-DPtrIntType=long -DCPU=I80586 -DTARGET=\"i86Win32VC60\" -DWIN32 -D_WINDOWS -D_WIN32_WINNT=0x0400 -DWIN32_LEAN_AND_MEAN /O2 /Zi /GX /MD -DNDEBUG -c
	Static Debug	-DPtrIntType=long -DCPU=I80586 -DTARGET=\"i86Win32VC60\" -DWIN32 -D_WINDOWS -D_WIN32_WINNT=0x0400 -DWIN32_LEAN_AND_MEAN /Od /ZI /GX /GZ /MTd -c
	Dynamic Debug	-DPtrIntType=long -DCPU=I80586 -DTARGET=\"i86Win32VC60\" -DWIN32 -D_WINDOWS -D_WIN32_WINNT=0x0400 -DWIN32_LEAN_AND_MEAN /Od /ZI /GX /GZ /MDd -c
i86Win32VC70	Static Release	-DPtrIntType=long -DCPU=I80586 -DTARGET=\"i86Win32VC70\" -DWIN32 -D_WINDOWS -D_WIN32_WINNT=0x0400 -DWIN32_LEAN_AND_MEAN /O2 /Zi /GX /MT -DNDEBUG -c
	Dynamic Release	-DPtrIntType=long -DCPU=I80586 -DTARGET=\"i86Win32VC70\" -DWIN32 -D_WINDOWS -D_WIN32_WINNT=0x0400 -DWIN32_LEAN_AND_MEAN /O2 /Zi /GX /MD -DNDEBUG -c
	Static Debug	-DPtrIntType=long -DCPU=I80586 -DTARGET=\"i86Win32VC70\" -DWIN32 -D_WINDOWS -D_WIN32_WINNT=0x0400 -DWIN32_LEAN_AND_MEAN /Od /ZI /GX /GZ /MTd -c
	Dynamic Debug	-DPtrIntType=long -DCPU=I80586 -DTARGET=\"i86Win32VC70\" -DWIN32 -D_WINDOWS -D_WIN32_WINNT=0x0400 -DWIN32_LEAN_AND_MEAN /Od /ZI /GX /GZ /MDd -c

Table 9.6 **Library-Creation Details for Windows Host Architectures**

RTI Architecture	Library Format	Compiler Flags Used by RTI
i86Win32VS2003	Static Release	-DPtrIntType=long -DCPU=I80586 -DTARGET=\"i86Win32VS2003\" -DWIN32 -D_WINDOWS -D_WIN32_WINNT=0x0400 -DWIN32_LEAN_AND_MEAN /O2 /Zi /GX /MT -DNDEBUG -c
	Dynamic Release	-DPtrIntType=long -DCPU=I80586 -DTARGET=\"i86Win32VS2003\" -DWIN32 -D_WINDOWS -D_WIN32_WINNT=0x0400 -DWIN32_LEAN_AND_MEAN /O2 /Zi /GX /MD -DNDEBUG -c
	Static Debug	-DPtrIntType=long -DCPU=I80586 -DTARGET=\"i86Win32VS2003\" -DWIN32 -D_WINDOWS -D_WIN32_WINNT=0x0400 -DWIN32_LEAN_AND_MEAN /Od /ZI /GX /GZ /MTd -c
	Dynamic Debug	-DPtrIntType=long -DCPU=I80586 -DTARGET=\"i86Win32VS2003\" -DWIN32 -D_WINDOWS -D_WIN32_WINNT=0x0400 -DWIN32_LEAN_AND_MEAN /Od /ZI /GX /GZ /MDd -c
i86Win32VS2005	Static Release	-DPtrIntType=long -DCSREAL_IS_FLOAT -DCPU=I80586 -DTARGET=\"i86Win32VS2005\" -DWIN32 -D_WINDOWS -D_WIN32_WINNT=0x0400 -DWIN32_LEAN_AND_MEAN /O2 /Zi /MT /EHsc -D_CRT_SECURE_NO_DEPRECATE -DNDEBUG -c
	Dynamic Release	-DPtrIntType=long -DCSREAL_IS_FLOAT -DCPU=I80586 -DTARGET=\"i86Win32VS2005\" -DWIN32 -D_WINDOWS -D_WIN32_WINNT=0x0400 -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=\"i86Win32VS2005\" -DWIN32 -D_WINDOWS -D_WIN32_WINNT=0x0400 -DWIN32_LEAN_AND_MEAN /Od /ZI /MTd /EHsc /RTC1 -D_CRT_SECURE_NO_DEPRECATE -c
	Dynamic Debug	-DPtrIntType=long -DCSREAL_IS_FLOAT -DCPU=I80586 -DTARGE T=\"i86Win32VS2005\" -DWIN32 -D_WINDOWS -D_WIN32_WINNT=0x0400 -DWIN32_LEAN_AND_MEAN /Od /ZI /MDd /EHsc /RTC1 -D_CRT_SECURE_NO_DEPRECATE -c

Table 9.6 **Library-Creation Details for Windows Host Architectures**

RTI Architecture	Library Format	Compiler Flags Used by RTI
i86Win32VS2008	Static Release	-DPtrIntType=long -DCSREAL_IS_FLOAT -DCPU=I80586 -DTARGET=\"i86Win32VS2008\" -DWIN32 -D_WINDOWS -D_WIN32_WINNT=0x0501 -DWIN32_LEAN_AND_MEAN /O2 /Zi /MT /EHsc -D_CRT_SECURE_NO_DEPRECATE -DNDEBUG -c
	Dynamic Release	-DPtrIntType=long -DCSREAL_IS_FLOAT -DCPU=I80586 -DTARGET=\"i86Win32VS2008\" -DWIN32 -D_WINDOWS -D_WIN32_WINNT=0x0501 -DWIN32_LEAN_AND_MEAN /O2 /Zi /MD /EHsc -D_CRT_SECURE_NO_DEPRECATE -DNDEBUG -c
	Static Debug	-DPtrIntType=long -DCSREAL_IS_FLOAT -DCPU=I80586 -DTARGET=\"i86Win32VS2008\" -DWIN32 -D_WINDOWS -D_WIN32_WINNT=0x0501 -DWIN32_LEAN_AND_MEAN /Od /ZI /MTd /EHsc /RTC1 -D_CRT_SECURE_NO_DEPRECATE -c
	Dynamic Debug	-DPtrIntType=long -DCSREAL_IS_FLOAT -DCPU=I80586 -DTARGE T=\"i86Win32VS2008\" -DWIN32 -D_WINDOWS -D_WIN32_WINNT=0x0501 -DWIN32_LEAN_AND_MEAN /Od /ZI /MDd /EHsc /RTC1 -D_CRT_SECURE_NO_DEPRECATE -c
i86Win32VS2010	Static Release	-DPtrIntType=long -DCSREAL_IS_FLOAT -DCPU=I80586 -DTARGET=\"i86Win32VS2010\" -DWIN32 -D_WINDOWS -D_WIN32_WINNT=0x0501 -DWIN32_LEAN_AND_MEAN /O2 /Zi /MT /EHsc -D_CRT_SECURE_NO_DEPRECATE -DNDEBUG -c
	Dynamic Release	-DPtrIntType=long -DCSREAL_IS_FLOAT -DCPU=I80586 -DTARGET=\"i86Win32VS2010\" -DWIN32 -D_WINDOWS -D_WIN32_WINNT=0x0501 -DWIN32_LEAN_AND_MEAN /O2 /Zi /MD /EHsc -D_CRT_SECURE_NO_DEPRECATE -DNDEBUG -c
	Static Debug	-DPtrIntType=long -DCSREAL_IS_FLOAT -DCPU=I80586 -DTARGET=\"i86Win32VS2010\" -DWIN32 -D_WINDOWS -D_WIN32_WINNT=0x0501 -DWIN32_LEAN_AND_MEAN /Od /ZI /MTd /EHsc /RTC1 -D_CRT_SECURE_NO_DEPRECATE -c
	Dynamic Debug	-DPtrIntType=long -DCSREAL_IS_FLOAT -DCPU=I80586 -DTARGE T=\"i86Win32VS2010\" -DWIN32 -D_WINDOWS -D_WIN32_WINNT=0x0501 -DWIN32_LEAN_AND_MEAN /Od /ZI /MDd /EHsc /RTC1 -D_CRT_SECURE_NO_DEPRECATE -c

Table 9.6 **Library-Creation Details for Windows Host Architectures**

RTI Architecture	Library Format	Compiler Flags Used by RTI
x64Win64dotnet2.0	Dynamic Release	/O2 /GL /D "WIN64" /D "NDEBUG" /D "RTI_WIN64" /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 "RTI_WIN64" /D "NDDS_DLL_VARIABLE" /D "_WINDLL" /D "_UNICODE" /D "UNICODE" /FD /EHa /MDd /c /Zi /clr /TP
x64Win64jdk	Dynamic Release	-target 1.6 –source 1.6
	Dynamic Debug	-target 1.6 –source 1.6 -g
x64Win64VS2005 Note: linker requires /MACHINE:X64 option.	Static Release	/W3 -DPtrIntType=long -DCSREAL_IS_FLOAT -DCPU=AMD64 -DTARGET=\"x64Win64VS2005\" -DWIN32 -D_WINDOWS -D_WIN32_WINNT=0x0400 -DWIN32_LEAN_AND_MEAN /O2 /Zi /MT /EHsc -D_CRT_SECURE_NO_DEPRECATE -DNDEBUG -c
	Dynamic Release	/W3 -DPtrIntType=long -DCSREAL_IS_FLOAT -DCPU=AMD64 -DTARGET=\"x64Win64VS2005\" -DWIN32 -D_WINDOWS -D_WIN32_WINNT=0x0400 -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=\"x64Win64VS2005\" -DWIN32 -D_WINDOWS -D_WIN32_WINNT=0x0400 -DWIN32_LEAN_AND_MEAN /Od /ZI /MTd /EHsc /RTC1 -D_CRT_SECURE_NO_DEPRECATE -c
	Dynamic Debug	/W3 -DPtrIntType=long -DCSREAL_IS_FLOAT -DCPU=AMD64 -DTARGET=\"x64Win64VS2005\" -DWIN32 -D_WINDOWS -D_WIN32_WINNT=0x0400 -DWIN32_LEAN_AND_MEAN /Od /ZI /MDd /EHsc /RTC1 -D_CRT_SECURE_NO_DEPRECATE -c

Table 9.6 **Library-Creation Details for Windows Host Architectures**

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

Table 9.7 **Library-Creation Details for Windows Target Architectures (Windows CE)**

RTI Architecture	Library Format	Compiler Flags Used by RTI
armv4WinCE6.0 VS2005	Static Release	-DPtrIntType=long -DCSREAL_IS_FLOAT -D CPU=ARMV4 -D TARGET=\"armv4WinCE6.0VS2005\" -D WIN32_PLATFORM_PSPC -D _WIN32_WCE=0x502 -D UNICODE -D _UNICODE -D _LIB -D _M_ARM=4 /O2 /Zi /MT /EHsc -D UNDER_CE=0x502 -D _VC80_UPGRADE=0x0600 /fp:fast /GR /GS- -DNDEBUG -c
	Static Debug	-DPtrIntType=long -DCSREAL_IS_FLOAT -D CPU=ARMV4 -D TARGET=\"armv4WinCE6.0VS2005\" -D WIN32_PLATFORM_PSPC -D _WIN32_WCE=0x502 -D UNICODE -D _UNICODE -D _LIB -D _M_ARM=4 /Od /Zi /MTd /EHsc -D UNDER_CE=0x502 -D _VC80_UPGRADE=0x0600 /fp:fast /GR /GS- -c
	Dynamic Release	-DPtrIntType=long -DCSREAL_IS_FLOAT -D CPU=ARMV4 -D TARGET=\"armv4WinCE6.0VS2005\" -D WIN32_PLATFORM_PSPC -D _WIN32_WCE=0x502 -D UNICODE -D _UNICODE -D _LIB -D _M_ARM=4 /O2 /Zi /MD /EHsc -D UNDER_CE=0x502 -D _VC80_UPGRADE=0x0600 /fp:fast /GR /GS- -DNDEBUG -c
	Dynamic Debug	-DPtrIntType=long -DCSREAL_IS_FLOAT -D CPU=ARMV4 -D TARGET=\"armv4WinCE6.0VS2005\" -D WIN32_PLATFORM_PSPC -D _WIN32_WCE=0x502 -D UNICODE -D _UNICODE -D _LIB -D _M_ARM=4 /Od /Zi /MDd /EHsc -D UNDER_CE=0x502 -D _VC80_UPGRADE=0x0600 /fp:fast /GR /GS- -c
i86WinCE6.0VS2005	Static Release	-DPtrIntType=long -DCSREAL_IS_FLOAT -D CPU=i586 -D TARGET=\"i86WinCE6.0eVS2005\" -D WIN32_PLATFORM_PSPC -D _WIN32_WCE=0x600 -D UNICODE -D _UNICODE -D _LIB -D _M_IX86=300 /O2 /Zi /MT /EHsc -D UNDER_CE=0x600 /GR -DNDEBUG -c
	Static Debug	-DPtrIntType=long -DCSREAL_IS_FLOAT -D CPU=i586 -D TARGET=\"i86WinCE6.0eVS2005\" -D WIN32_PLATFORM_PSPC -D _WIN32_WCE=0x600 -D UNICODE -D _UNICODE -D _LIB -D _M_IX86=300 /Od /Zi /MTd /EHsc -D UNDER_CE=0x600 /GR -c
	Dynamic Release	-DPtrIntType=long -DCSREAL_IS_FLOAT -D CPU=i586 -D TARGET=\"i86WinCE6.0eVS2005\" -D WIN32_PLATFORM_PSPC -D _WIN32_WCE=0x600 -D UNICODE -D _UNICODE -D _LIB /MD /EHsc -D UNDER_CE=0x600 /GR -DNDEBUG -c
	Dynamic Debug	-DPtrIntType=long -DCSREAL_IS_FLOAT -D CPU=i586 -D TARGET=\"i86WinCE6.0VS2005\" -D WIN32_PLATFORM_PSPC -D _WIN32_WCE=0x600 -D UNICODE -D _UNICODE -D _LIB -D _M_IX86=300 /Od /Zi /MDd /EHsc

Table 9.8 **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	

a. These libraries are located in <*wan install dir*>\lib\<*architecture*>\ (where <*wan install dir*> is where RTI Secure WAN Transport is installed, such as c:\rti\ndds.4.5x)

b. These libraries are located in <*openssl install dir*>\<*architecture*>\lib, where <*openssl install dir*> is where you installed OpenSSL, such as c:\rti\openssl-0.9.8f.

Table 9.9 **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

a. The libraries are located in <*DDS install dir*>\lib\<*architecture*>\, where <*DDS install dir*> is where you installed *RTI Data Distribution Service*, such as /local/rti/ndds.4.5x.

Table 9.10 **Additional Libraries for using RTI TCP Transport APIs on Windows Systems with TLS Enabled**

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

a. The libraries are located in <TLS install dir>/lib/<architecture>/., where <TLS install dir> is where you installed RTI TLS Support, such as /local/rti/ndds.4.5x.

10 Custom Supported Platforms

This section describes additional target libraries available with *RTI Data Distribution Service 4.5c*, for which RTI offers custom support. If you are interested in using one of these platforms, please contact your local RTI representative or email **sales@rti.com**.

Table 10.1 lists the custom supported Linux platforms.

Table 10.1 **Custom Supported Linux Platforms**

Operating System	CPU	Compiler	RTI Architecture Abbreviation
Mistral Linux Kernel 2.6.32	ARMv7	Sourcery G++ Lite 2009q3-67 gcc 4.4.1	armv7leLinux2.6gcc4.4.1
Red Hat Enterprise Linux 5.1	x86	gcc3.4.6	i86Linux2.6gcc3.4.6
		Sun Java Platform Standard Edition JDK 1.5 and 1.6	i86Linux2.6gcc3.4.6jdk
RedHawk Linux 5.1	x86	gcc 4.1.2	i86RedHawk5.1gcc4.1.2
		Sun Java Platform Standard Edition JDK 1.5 and 1.6	i86RedHawk5.1gcc4.1.2jdk