RTI Connext

Core Libraries and Utilities

Release Notes

Version 5.0.0

Your systems. Working as one.

Printed in U.S.A. First printing.
August 2012.

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

Contents

Release Notes

This document includes the following sections:

- ❏ System Requirements (Section 1)
- ❏ Compatibility (Section 2)
- ❏ What's Fixed in Connext 5.0.0 (Section 3)
- ❏ Known Issues (Section 4)
- ❏ Custom Supported Platforms (Section 5)
- ❏ Experimental Features (Section 6)

For an overview of new features, please see the *What's New* document (**RTI_CoreLibrariesAndUtilities_WhatsNew.pdf**).

For more information, visit the RTI Knowledge Base, accessible from https://support.rti.com/, to see sample code, general information on *RTI® Connext™* (formerly *RTI Data Distribution Service*), performance information, troubleshooting tips, and technical details. By its very nature, the knowledge base is continuously evolving and improving. We hope that you will find it helpful. If there are questions that you would like to see addressed or comments you would like to share, please send e-mail to **support@rti.com**. We can only guarantee a response to customers with a current maintenance contract or subscription. You can purchase a maintenance contract or subscription by contacting your local RTI representative (see http://www.rti.com/company/contact.html), sending an e-mail request to **sales@rti.com**, or calling +1 (408) 990-7400.

---

1 System Requirements

1.1 Supported Operating Systems

RTI Connext requires a multi-threaded operating system. This section describes the host and target systems supported by *Connext*.

In this context, a *host* is the computer on which you will be developing a *Connext* application. A *target* is the computer on which the completed application will run. A host installation provides the code generation tool (*rtiddsgen*), examples and documentation, as well as the header files required to build a *Connext* application for any architecture. You will also need a target installation, which provides the libraries required to build a *Connext* application for that particular target architecture.

Table 1.0 lists the platforms available with *Connext* 5.0.0.

Table 1.0 **Platforms Available with Connext 5.0.0**

Platform	Operating System	Reference
AIX®	AIX 5.3	Table 1.1 on page 4
INTEGRITY® (target only)	INTEGRITY 5.0.11, 10.0.2	Table 1.2 on page 4
Linux® (Cell BE™)	Fedora™ 12 (2.6.32 kernel)	Table 1.3 on page 5
Linux (Intel®)	CentOS 5.4, 5.5, 6.0 Fedora 12 (2.6.32 kernel) Fedora 12 (2.6.32 kernel) with gcc 4.5.1 Red Hat® Enterprise Linux 5.0-5.2, 5.4, 5.5, 6.0, 6.1 Red Hat Enterprise Linux 5.2 with Real-Time Extensions SUSE® Linux Enterprise Server 10.1 (2.6 kernel) Ubuntu® Server 10.04 (2.6 kernel) Wind River® Linux 4 (2.6 kernel)	Table 1.4 on page 5
Linux (PowerPC®) (target only)	Freescale™ P2020RDB (2.6.32 kernel) SELinux (2.6.32 kernel) Wind River® Linux 3 Yellow Dog™ Linux 4.0	Table 1.5 on page 6
LynxOS® (target only)[1]	LynxOS 4.0, 4.2, 5.0	Table 1.6 on page 6
Mac OS®	Mac OS X 10.6	Table 1.7 on page 6
QNX® (target only)	QNX Neutrino® 6.4.1, 6.5	Table 1.8 on page 6
Solaris™	Solaris 2.9, 2.10	Table 1.9 on page 7
VxWorks® (target only)	VxWorks 5.5, 6.3 - 6.9	Table 1.10 on page 7
VxWorks 653 (target only)	VxWorks 653 2.3	
VxWorks MILS (target only)	VxWorks MILS 2.1.1	
Windows®	Windows 7 (32-bit and 64-bit Editions) Windows 2003 (32-bit and 64-bit Editions) Windows Server 2008 R2 (64-bit Edition) Windows Vista® (32-bit and 64-bit Editions) Windows XP Professional (32-bit and 64-bit Editions) with Service Pack 2	Table 1.11 on page 8

1. The Java API is not supported on LynxOS platforms in 5.0.0. If your application requires support for Java on LynxOS, please contact your RTI account manager.

Visual Studio® 2005 — Service Pack 1 Redistributable Package MFC Security Update is Required

❏ You must have the Microsoft Visual C++ 2005 Service Pack 1 Redistributable Package MFC Security Update installed on the machine where you are *running* an application built with the release or debug libraries of the following RTI architecture packages:

- i86Win32VS2005 and x64Win64VS2005, built with dynamic libraries
- i86Win32jdk and x64Win64jdk
- i86Win32dotnet2.0 and x64Win64dotnet2.0

The Microsoft Visual C++ 2005 Service Pack 1 Redistributable Package MFC Security Update can be obtained from the following Microsoft website:

- http://www.microsoft.com/download/en/details.aspx?id=26347

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 release libraries of the following RTI architecture packages:

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

To run an application built with *debug* libraries of the above RTI architecture packages, you must have Visual Studio 2008 Service Pack 1 installed.

The Microsoft Visual C++ 2008 Service Pack 1 Redistribution Package can be 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

Visual Studio 2010 - Service Pack 1 Requirement

❏ You must have Visual Studio 2010 Service Pack 1 or the Microsoft Visual C++ 2010 SP1 Redistribution Package installed on the machine where you are *running* an application built with the release libraries of the following RTI architecture packages:

- i86Win32VS2010 built with dynamic libraries
- x64Win64VS2010 built with dynamic libraries
- i86Win32dotnet4.0 and x64Win64dotnet4.0

To run an application built with *debug* libraries of the above RTI architecture packages, you must have Visual Studio 2010 Service Pack 1 installed.

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

- For x86 architectures:

  http://www.microsoft.com/download/en/details.aspx?id=5555

- For x64 architectures:

  http://www.microsoft.com/download/en/details.aspx?id=14632

Note: Additional platforms not listed in this document may be supported through special development and maintenance agreements. Contact your RTI sales representative for details.

The following tables provide additional details. See the *RTI Core Libraries and Utilities User's Manual* and *Platform Notes* for more information on compilers and linkers.

❏ Table 1.1, "AIX Platforms," on page 1-4

❏ Table 1.2, "INTEGRITY Platforms," on page 1-4

❏ Table 1.3, "Linux Platforms on Cell BE CPUs," on page 1-5

❏ Table 1.4, "Linux Platforms on Intel CPUs," on page 1-5

❏ Table 1.5, "Linux Platforms on PowerPC CPUs," on page 1-6

❏ Table 1.6, "LynxOS Platforms," on page 1-6

❏ Table 1.7, "Mac OS Platforms," on page 1-6

1.2 Disk and Memory Usage

Disk usage for a typical host-only installation is approximately 250 MB. Each additional architecture (host or target) requires an additional 75 MB.

We recommend that you have at least 256 MB RAM installed on your host development system. The target requirements are significantly smaller and they depend on the complexity of your application and hardware architecture.

1.3 Networking Support

Connext includes full support for pluggable transports. *Connext* applications can run over various communication media, such as UDP/IP over Ethernet, and local inter-process shared memory—provided the correct transport plug-ins for the media are installed.

By default, *Connext* uses the UDP/IPv4 and shared-memory transport plug-ins. The shared memory transport is not supported for VxWorks 5.5.

A built-in IPv6 transport is also available (disabled by default) for these platforms:

- Linux/Fedora: All platforms except SELinux (2.6.32. kernel)
- QNX: All platforms
- Solaris: All platforms
- VxWorks 6.7 (except ppc405Vx6.6gcc4.1.2), 6.8, 6.9
- Windows: All platforms

A TCP transport is also available (but is not a *built-in* transport) for the following platforms:

- Red Hat Enterprise Linux 5.0-5.2, 5.4, 5.5, 6.0, 6.1; CentOS 5.4, 5.5, 6.0; and Ubuntu 10.04
- Windows with Visual Studio 2005, 2008, and 2010.

Supported architectures appear on the following pages, followed by Compatibility (Section 2).

Table 1.1 **AIX 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 1.2 **INTEGRITY Platforms**

Operating System	CPU	Compiler	IP Stack	RTI Architecture Abbreviation
INTEGRITY 5.0.11	PPC 85xx	multi 4.2.4	GHnet2 IP stack[1]	ppc85xxInty5.0.11.xes-p2020
INTEGRITY 10.0.2[2]	x86	multi 5.0.6	CHNet IPv4 stack	pentiumInty10.0.2.pcx86

1. Kernel must be built using -lip4 or -lip46.
2. Requires patch_6901.iff from Green Hills Software.

Table 1.3 **Linux Platforms on Cell BE CPUs**

Operating System	CPU	Compiler	RTI Architecture Abbreviation
Fedora 12 (2.6.32 kernel)	Cell BE	gcc 4.5.1[1], glib 2.9	cell64Linux2.6gcc4.5.1

1. Requires a custom version of gcc 4.5.1.

Table 1.4 **Linux Platforms on Intel CPUs**

Operating System	CPU	Compiler	RTI Architecture Abbreviation
CentOS 5.4, 5.5 (2.6 kernel)	x86	gcc 4.1.2	i86Linux2.6gcc4.1.2
		Sun Java Platform Standard Edition JDK 1.6	i86Linux2.6gcc4.1.2jdk
	x64	gcc 4.1.2	x64Linux2.6gcc4.1.2
		Sun Java Platform Standard Edition JDK 1.6	x64Linux2.6gcc4.1.2jdk
CentOS 6.0	x86	gcc 4.4.5	i86Linux2.6gcc4.4.5
		Sun Java Platform Standard Edition JDK 1.6	i86Linux2.6gcc4.4.5jdk
	x64	gcc 4.4.5	x64Linux2.6gcc4.4.5
		Sun Java Platform Standard Edition JDK 1.6	x64Linux2.6gcc4.4.5jdk
Fedora 12 (2.6 kernel)	x64	gcc 4.4.4	x64Linux2.6gcc4.4.4
Fedora 12 (2.6.32 kernel) with gcc 4.5.1	x64	gcc 4.5.1[1]	x64Linux2.6gcc4.5.1
Red Hat Enterprise Linux 5.0 (2.6 kernel)	x86	gcc 4.1.1	i86Linux2.6gcc4.1.1
		Sun Java Platform Standard Edition JDK 1.6	i86Linux2.6gcc4.1.1jdk
	x64	gcc 4.1.1	x64Linux2.6gcc4.1.1
		Sun Java Platform Standard Edition JDK 1.6	x64Linux2.6gcc4.1.1jdk
Red Hat Enterprise Linux 5.1, 5.2, 5.4, 5.5 (2.6 kernel)	x86	gcc 4.1.2	i86Linux2.6gcc4.1.2
		Sun Java Platform Standard Edition JDK 1.6	i86Linux2.6gcc4.1.2jdk
	x64	gcc 4.1.2	x64Linux2.6gcc4.1.2
		Sun Java Platform Standard Edition JDK 1.6	x64Linux2.6gcc4.1.2jdk
Red Hat Enterprise Linux 5.2 with Real-Time Extensions (2.6 kernel)	x86	gcc 4.1.2	i86Linux2.6gcc4.1.2
		Sun Java Platform Standard Edition JDK 1.6	i86Linux2.6gcc4.1.2jdk
Red Hat Enterprise Linux 6.0, 6.1 (2.6 kernel)	x86	gcc 4.4.5	i86Linux2.6gcc4.4.5
	x64	gcc 4.4.5	x64Linux2.6gcc4.4.5
SUSE Linux Enterprise Server 10.1 (2.6 kernel)	x86	gcc 4.1.0	i86Suse10.1gcc4.1.0
		Sun Java Platform Standard Edition JDK 1.5 and 1.6	i86Suse10.1gcc4.1.0jdk
	x64	gcc 4.1.0	x64Suse10.1gcc4.1.0
		Sun Java Platform Standard Edition JDK 1.5 and 1.6	x64Suse10.1gcc4.1.0jdk

Table 1.4 **Linux Platforms on Intel CPUs**

Operating System	CPU	Compiler	RTI Architecture Abbreviation
Ubuntu Server 10.04 (LTS)	x86	gcc 4.4.3	i86Linux2.6gcc4.4.3
		Sun Java Platform Standard Edition JDK 1.6	i86Linux2.6gcc4.4.3jdk
	x64	gcc 4.4.3	x64Linux2.6gcc4.4.3
		Sun Java Platform Standard Edition JDK 1.6	x64Linux2.6gcc4.4.3jdk
Wind River Linux 4 (2.6 kernel)	x64	gcc 4.4.1	x64WRLinux2.6gcc4.4.1

1. Requires a custom version of gcc 4.5.1.

Table 1.5 **Linux Platforms on PowerPC CPUs**

Operating System	CPU	Compiler	RTI Architecture Abbreviation
Freescale P2020RDB (2.6.32 kernel)	PPC 85xx	Freescale gcc.4.3.74 based on gcc.4.3.2	ppc85xxLinux2.6gcc4.3.2
SELinux (2.6.32 kernel)	PPC 4xxFP	gcc 4.5.1[1], glibc 2.9	ppc4xxFPLinux2.6gcc4.5.1
Wind River Linux 3	PPC 85xx	gcc 4.3.2	ppc85xxWRLinux2.6gcc4.3.2
Yellow Dog® Linux 4.0 (2.6 kernel)	PPC 74xx (such as 7410)	gcc 3.3.3	ppc7400Linux2.6gcc3.3.3

1. Requires a custom version of gcc 4.5.1.

Table 1.6 **LynxOS Platforms**

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

Table 1.7 **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 1.8 **QNX Platforms**

Operating System	CPU	Compiler	RTI Architecture Abbreviation
QNX Neutrino 6.4.1	x86	qcc 4.3.3 with GNU C++ libraries	i86QNX6.4.1qcc_gpp
QNX Neutrino 6.5	x86	qcc 4.4.2 with GNU C++ libraries	i86QNX6.5qcc_gpp4.4.2

Table 1.9 **Solaris Platforms**

Operating System	CPU	Compiler	RTI Architecture Abbreviation
Solaris 2.9	x86	gcc 3.3.2	i86Sol2.9gcc3.3.2
		Sun Java Platform Standard Edition JDK 1.6	i86Sol2.9jdk
	UltraSPARC	CC 5.4 (Forte Dev 7, Sun One Studio 7)	sparcSol2.9cc5.4
		Sun Java Platform Standard Edition JDK 1.6	sparcSol2.9jdk
Solaris 2.10	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)	gcc3.4.2	sparc64Sol2.10gcc3.4.2
		Sun Java Platform Standard Edition JDK 1.5 or 1.6	sparc64Sol2.10jdk

Table 1.10 **VxWorks Target Platforms**[1]

Operating System	CPU	Compiler	RTI Architecture
VxWorks 5.5	PPC 603	gcc 2.96	ppc603Vx5.5gcc
	PPC 604	gcc 2.96	ppc604Vx5.5gcc
	PPC 750	gcc 2.96	ppc603Vx5.5gcc
	PPC 7400	gcc 2.96	ppc603Vx5.5gcc
VxWorks 6.3, 6.4	Any Wind River PPC32 CPU with floating point hardware	gcc 3.4.4	For kernel modules: ppc604Vx6.3gcc3.4.4 For Real Time Processes: ppc604Vx6.3gcc3.4.4_rtp
VxWorks 6.5	Any Wind River PPC32 CPU with floating point hardware	gcc 3.4.4	For kernel modules: ppc604Vx6.5gcc3.4.4 For Real Time Processes: ppc604Vx6.5gcc3.4.4_rtp
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
	PPC 405[2]	gcc 4.1.2	For Kernel Modules: ppc405Vx6.6gcc4.1.2 For Real Time Processes: ppc405Vx6.6gcc4.1.2_rtp

Table 1.10 **VxWorks Target Platforms**[1]

Operating System	CPU	Compiler	RTI Architecture
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 on non-SMP systems: ppc604Vx6.7gcc4.1.2_rtp For Real Time Processes on SMP systems: ppc604Vx6.7gcc4.1.2_smp
	PPC 405[2]	gcc 4.1.2	For Kernel Modules: ppc405Vx6.7gcc4.1.2 For Real Time Processes: ppc405Vx6.7gcc4.1.2_rtp
VxWorks 6.8	Pentium	gcc 4.1.2	For Kernel Modules: pentiumVx6.8gcc4.1.2 For Real Time Processes: pentiumVx6.8gcc4.1.2_rtp
	Any Wind River PPC32 CPU with floating point hardware	gcc 4.1.2	For Kernel Modules: ppc604Vx6.8gcc4.1.2 For Real Time Processes on a non-SMP system: ppc604Vx6.8gcc4.1.2_rtp
VxWorks 6.9	Pentium32-bit	gcc 4.3.3	For Kernel Modules: pentiumVx6.9gcc4.3.3 For Real Time Processes: pentiumVx6.9gcc4.3.3_rtp
	Any Wind River PPC32 CPU with floating point hardware	gcc 4.3.3	For Kernel Modules: ppc604Vx6.9gcc4.3.3 For Real Time Processes: ppc604Vx6.9gcc4.3.3_rtp
VxWorks 653 2.3	sbc8641d	gcc 3.32	sbc8641Vx653-2.3gcc3.3.2
	SIMPC	gcc 3.32	simpcVx653-2.3gcc3.3.2
VxWorks MILS 2.1.1 with vThreads 2.2.3	ppc85xx	gcc 3.3.2	ppc85xxVxT2.2.3gcc3.3.2

1. For use with Windows and/or Solaris Hosts as supported by Wind River Systems.
2. For ppc405, the architecture string is the same for VxWorks 6.6 and 6.7.

Table 1.11 **Windows Platforms**

Operating System	CPU	Compiler or Software Development Kit [1 2]	RTI Architecture
Windows 7 (32-bit Edition)	x86	Visual Studio 2010	i86Win32VS2010
		Visual Studio 2010 (C++/CLI, C# 8.0 or 9.0)	i86Win32dotnet4.0
		Sun Java Platform Standard Edition JDK 1.6	i86Win32jdk

Table 1.11 **Windows Platforms**

Operating System	CPU	Compiler or Software Development Kit [1] [2]	RTI Architecture
Windows 7 (64-bit Edition)	x64	Visual Studio 2010	x64Win64VS2010
		Visual Studio 2010 (C++/CLI, C# 8.0 or 9.0)	x64Win64dotnet4.0
		Sun Java Platform Standard Edition JDK 1.6	x64Win64jdk
Windows 2003 (32-bit Edition)	x86	Visual Studio 2005 SP 1	i86Win32VS2005
		Visual Studio 2005 SP 1 (C++/CLI, C# 8.0 or 9.0)	i86Win32dotnet2.0
		Visual Studio 2008 SP 1	i86Win32VS2008
		Visual Studio 2008 SP 1 (C++/CLI, C# 8.0 or 9.0)	i86Win32dotnet2.0
		Sun Java Platform Standard Edition JDK 1.6	i86Win32jdk
Windows 2003 (64-bit Edition)	x64	Visual Studio 2005 SP 1	x64Win64VS2005
		Visual Studio 2005 SP 1 (C++/CLI, C# 8.0 or 9.0)	x64Win64dotnet2.0
		Visual Studio 2008 SP 1	x64Win64VS2008
		Sun Java Platform Standard Edition JDK 1.6	x64Win64jdk
Windows Server 2008 R2 (64-bit Edition)	x64	Visual Studio 2005 SP 1 (C++/CLI, C# 8.0 or 9.0)	x64Win64dotnet2.0
		Visual Studio 2010	x64Win64VS2010
		Visual Studio 2010 (C++/CLI, C# 8.0 or 9.0)	x64Win64dotnet4.0
		Sun Java Platform Standard Edition JDK 1.6	x64Win64jdk
Windows Vista (32-bit Edition)	x86	Visual Studio 2005 SP 1	i86Win32VS2005
		Visual Studio 2005 SP 1 (C++/CLI, C# 8.0 or 9.0)	i86Win32dotnet2.0
		Visual Studio 2008 SP 1	i86Win32VS2008
		Visual Studio 2008 SP 1 (C++/CLI, C# 8.0 or 9.0)	i86Win32dotnet2.0
		Sun Java Platform Standard Edition JDK 1.6	i86Win32jdk
Windows Vista (64-bit Edition)	x64	Visual Studio 2005 SP 1	x64Win64VS2005
		Visual Studio 2005 SP 1 (C++/CLI, C# 8.0 or 9.0)	x64Win64dotnet2.0
		Visual Studio 2008 SP1	x64Win64VS2008
		Visual Studio 2008 SP 1 (C++/CLI, C# 8.0 or 9.0)	x64Win32dotnet2.0
		Sun Java Platform Standard Edition JDK 1.6	x64Win64jdk
Windows XP Professional[3] (32-bit Edition) Service Pack 2	x86	Visual Studio 2005 SP 1	i86Win32VS2005
		Visual Studio 2005 SP 1 (C++/CLI, C# 8.0 or 9.0)	i86Win32dotnet2.0
		Visual Studio 2008 SP 1	i86Win32VS2008
		Visual Studio 2008 SP 1 (C++/CLI, C# 8.0 or 9.0)	i86Win32dotnet2.0
		Sun Java Platform Standard Edition JDK 1.6	i86Win32jdk
Windows XP Professional (64-bit Edition) Service Pack 2	x64	Visual Studio 2005 SP 1	x64Win64VS2005
		Visual Studio 2005 SP 1 (C++/CLI, C# 8.0 or 9.0)	x64Win64dotnet2.0
		Visual Studio 2008 SP 1	x64Win64VS2008
		Visual Studio 2008 SP 1 (C++/CLI, C# 8.0 or 9.0)	x64Win32dotnet2.0
		Sun Java Platform Standard Edition JDK 1.6	x64Win64jdk

1. 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.)

2. 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#.
3. 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.

2 Compatibility

RTI strives to provide a seamless upgrade path when the product is updated. When upgrading to a new version of *Connext*, there are five components to consider:

- Wire Protocol Compatibility (Section 2.1)
- Code Compatibility (Section 2.2)
- Extensible Types Compatibility (Section 2.3)
- ODBC Database Compatibility (Section 2.4)
- Transport Compatibility (Section 2.5)

2.1 Wire Protocol Compatibility

2.1.1 General Information on RTPS (All Releases)

Connext communicates over the wire using the formal Real-time Publish-Subscribe (RTPS) protocol. RTPS has been developed from the ground up with performance, interoperability and extensibility in mind. The RTPS protocol is an international standard managed by the OMG. The RTPS protocol has built-in extensibility mechanisms that enable new revisions to introduce new message types, extend the existing messages, or extend the Quality of Service settings in the product—without breaking interoperability.

RTPS 1.0 was introduced in 2001. The current version is 2.1. RTI plans to maintain interoperability between middleware versions based on RTPS 2.x.

2.1.2 Release-Specific Information for Connext 5.0

2.1.2.1 Large Data with Endpoint Discovery

An endpoint (*DataWriter* or *DataReader*) created with *Connext* 5.0 will not be discovered by an application that uses a previous *Connext* release if any of this conditions are met:

- The endpoint's TypeObject is sent on the wire *and* its size is greater than 65535 bytes. For additional information on TypeObjects, see the new *Core Libraries and Utilities Getting Started Guide Addendum for Extensible Types* (**RTI_CoreLibrariesAndUtilities_GettingStarted_ExtensibleTypesAddendum.pdf**).
- The endpoint's UserDataQosPolicy value is greater than 65535 bytes.

TypeObjects and UserDataQosPolicy values with a serialized size greater than 65535 bytes require extended parameterized encapsulation when they are sent as part of the endpoint discovery information. This parameterized encapsulation is not understood by previous *Connext* releases.

2.1.3 Release-Specific Information for Connext 4.5 and 5.0

Connext 4.5 and 5.0 are compatible with *RTI Data Distribution Service* 4.2 - 4.5, except as noted below.

2.1.3.1 RTPS Versions

Connext 4.5 and 5.0 support RTPS 2.1. Earlier releases (*RTI Data Distribution Service* 4.2c and lower) supported RTPS 1. Because the two RTPS versions are incompatible with each other, applications built with *Connext* 4.5 and 5.0 (and *RTI Data Distribution Service* 4.2e and higher), will not interoperate with applications built using *RTI Data Distribution Service* 4.2c or lower.

2.1.3.2 double, long long, unsigned long long or long double Wire Compatibility

If your *Connext* application's data type uses a 'double,' 'long long,' 'unsigned long long,' or 'long double,' it will not interoperate with applications built with *RTI Data Distribution Service* 4.2e or lower, unless you use the **-use42eAlignment** flag when generating code with *rtiddsgen*.

2.1.3.3 Sending 'Large Data' between RTI Data Distribution Service4.4d and Older Releases

The 'large data' format in *RTI Data Distribution Service* 4.2e, 4.3, 4.4b and 4.4c is not compliant with RTPS 2.1. ('Large data' refers to data that cannot be sent as a single packet by the transport.)

This issue is resolved in *Connext* and in *RTI Data Distribution Service* 4.4d-4.5e. As a result, by default, large data in *Connext* and in *RTI Data Distribution Service* 4.4d-4.5e is not compatible with older versions of *RTI Data Distribution Service*. You can achieve backward compatibility by setting the following properties to 1.

```
dds.data_writer.protocol.use_43_large_data_format
dds.data_reader.protocol.use_43_large_data_format
```

The properties can be set per *DataWriter/DataReader* or per *DomainParticipant*.

For example:

```
<participant_qos>
    <property>
        <value>
            <element>
                <name>
                    dds.data_writer.protocol.use_43_large_data_format
                </name>
                <value>1</value>
            </element>
            <element>
                <name>
                    dds.data_reader.protocol.use_43_large_data_format
                </name>
                <value>1</value>
            </element>
        </value>
   </participant_qos>
```

2.2 Code Compatibility

2.2.1 General Information (All Releases)

Connext uses an API that is an extension of the OMG Data Distribution Service (DDS) standard API. RTI strives to maintain API compatibility between versions, but will conform to changes in the OMG DDS standard.

Connext primarily consists of a library and a set of header files. In most cases, upgrading simply requires you to recompile your source using the new header files and link the new libraries. In some cases, minor modifications to your application code might be required; any such changes are noted in this document.

RTI allows you to define the data types that will be used to send and receive messages. To create code for a data type, *Connext* includes a tool called *rtiddsgen*. For input, *rtiddsgen* takes a data-type description (in IDL, XML, XSD, or WSDL format); *rtiddsgen* generates header files (or a class in Java) that can be used to send and receive data of the defined type. It also generates code that takes care of low-level details such as transforming the data into a machine-independent representation suitable for communication.

While this is not the common case, some upgrades require you to regenerate the code produced by *rtiddsgen*. The regeneration process is very simple; you only need to run the new version of *rtiddsgen* using the original input IDL file. This process will regenerate the header and source files, which can then be compiled along with the rest of your application.

2.2.2 Release-Specific Information for Connext 5.0

This section points out important differences in *Connext* 5.0 that may require changes on your part when upgrading to this release.

2.2.2.1 Required Change for Building with C++ Libraries for QNX Platforms—New in 5.0.0

For QNX architectures, starting in release 5.0.0: The C++ libraries are now built *without* the **-fno-rtti** flag and *with* the **-fexceptions** flag. To build QNX architectures starting with release 5.0.0, you must build your C++ applications *without* **-fno-exceptions** in order to link with the RTI libraries. In summary:

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

2.2.2.2 Changes to the Custom Content Filters API

In *Connext* 5.0, the ContentFilter's 'evaluate' function now receives a new 'struct DDS_FilterSampleInfo *' parameter that allows it to filter on meta-data.

The 'evaluate' function of previous custom filter implementations must be updated to add this new parameter.

2.2.2.3 Changes in Generated Type Support Code

The *rtiddsgen*-generated type-support code for user-defined data type changed in 5.0 to facilitate some new features. If you have code that was generated with *rtiddsgen* 4.5 or lower, you must regenerate that code using the version of *rtiddsgen* provided with this release.

2.2.3 Release-Specific Information for Connext 4.5 and 5.0

2.2.3.1 Type Support and Generated Code Compatibility

- **long long Native Data Type Support**

  In *Connext* (and *RTI Data Distribution Service* 4.5c,d,e), we assume all platforms natively support the 'long long' data type. This was not the case in older versions of *RTI Data Distribution Service*. There is no longer a need to define RTI_CDR_SIZEOF_LONG_LONG to be 8 on some platforms in order to map the DDS 'long long' data type to a native 'long long' type.

❑ **double, long long and unsigned long long Code Generation**

If your *Connext* (or *RTI Data Distribution Service* 4.3-4.5e) application's data type uses a 'double,' 'long long,' 'unsigned long long,' or 'long double,' it will not be backwards compatible with applications built with *RTI Data Distribution Service* 4.2e or lower, unless you use the **-use42eAlignment** flag when generating code with *rtiddsgen*.

❑ **Changes in Generated Type Support Code**

The *rtiddsgen*-generated type-support code for user-defined data type changed in 4.5 to facilitate some new features. If you have code that was generated using *rtiddsgen* 4.4 or lower, you must regenerate that code using the version of *rtiddsgen* provided with this release.

❑ **Cross-Language Instance Lookup when Using Keyed Data Types**

This issue only impacts systems using *RTI Data Distribution Service* 4.3.

In *RTI Data Distribution Service* 4.3, keys were serialized with the incorrect byte order when using the Java and .NET[1] APIs for the user-defined data type, resulting in incorrect behavior in the **lookup_instance()** and **get_key()** methods when using keyed data-types to communicate between applications in these languages and other programming languages. This issue was resolved in Java starting in *RTI Data Distribution Service* 4.3e rev. 01 and starting in .NET in *RTI Data Distribution Service* 4.4b.

As a result of this change, systems using keyed data that incorporate Java or .NET applications using both *RTI Data Distribution Service* 4.3 and this *Connext* release could experience problems in the **lookup_instance()** and **get_key()** methods. If you are affected by this limitation, please contact RTI Support.

❑ **Data Type with Variable-Size Keys**

If your data type contains more than one key field and at least one of the key fields *except the last one* is of variable size (for example, if you use a string followed by a long as the key):

- *RTI Data Distribution Service* 4.3e, 4.4b or 4.4c *DataWriters* may not be compatible with *RTI Data Distribution Service* 4.4d or higher *DataReaders*.
- *RTI Data Distribution Service* 4.3e, 4.4b or 4.4c *DataReaders* may not be compatible with *RTI Data Distribution Service* 4.4d or higher *DataWriters*.

Specifically, all samples will be received in those cases, but you may experience the following problems:

- Samples with the same key may be identified as different instances. (For the case in which the *DataWriter* uses *RTI Data Distribution Service* 4.4d-4.5e or *Connext*, this can only occur if the *DataWriter's* **disable_inline_keyhash** field (in the DataWriterProtocolQosPolicy) is true (this is not the default case).
- Calling **lookup_instance()** on the *DataReader* may return HANDLE_NIL even if the instance exists.

Please note that you probably would have had the same problem with this kind of data type already, even if both your *DataWriter* and *DataReader* were built with *RTI Data Distribution Service* 4.3e, 4.4b or 4.4c.

1. *RTI Connext* .NET language binding is currently supported for C# and C++/CLI.

If you are using a C/C++ or Java IDL type that belongs to this data type category in your *RTI Data Distribution Service* 4.3e, 4.4b or 4.4c application, you can resolve the backwards compatibility problem by regenerating the code with version of *rtiddsgen* distributed with *RTI Data Distribution Service* 4.4d. You can also upgrade your whole system to this release.

2.2.3.2 Other API and Behavior Changes

❑ **Code Compatibility Issue in C++ Applications using Dynamic Data**

If you are upgrading from a release prior to 4.5f and use Dynamic Data in a C++ application, you may need to make a minor code change to avoid a compilation error.

The error would be similar to this:

```
MyFile.cpp:1060: warning: extended initializer lists only available with
-std=c++0x or -std=gnu++0x
MyFile.cpp:1060: warning: extended initializer lists only available with
-std=c++0x or -std=gnu++0x
MyFile.cpp:1060: error: could not convert '{0l, 65536l, 1024l}' to
'DDS_DynamicDataProperty_t'
MyFile.cpp:1060: error: could not convert '{0u, 4294967295u,
4294967295u, 0u}' to 'DDS_DynamicDataTypeSerializationProperty_t
```

The code change involves using a constructor instead of a static initializer. Therefore if you have code like this:

```
DDS_DynamicDataTypeProperty_t properties =
    DDS_DynamicDataTypeProperty_t_INITIALIZER;
...
typeSupport = new DDSDynamicDataTypeSupport(typeCode, properties);
```

Replace the above with this:

```
DDS_DynamicDataTypeProperty_t properties;
...
typeSupport = new DDSDynamicDataTypeSupport(typeCode, properties);
```

❑ **New on_instance_replaced() method on DataWriterListener**

Starting with *RTI Data Distribution Service* 4.5c (and thereby included in *Connext*), there is a new DataWriterListener method, **on_instance_replaced()**, which supports the new instance replacement feature. This method provides notification that the maximum instances have been used and need to be replaced. If you are using a DataWriterListener from an older release, you may need to add this new method to your listener.

❑ **Counts in Cache Status and Protocol Status changed from Long to Long Long**

Starting with *RTI Data Distribution Service* 4.5c (and thereby included in *Connext*), all the 'count' data types in DataReaderCacheStatus, DataReaderProtocolStatus, DataWriterCacheStatus and DataWriterProtocolStatus changed from 'long' to 'long long' in the C, C++ and .NET[1] APIs in order to report the correct value for *Connext* applications that run for very long periods of time. If you have an application written with a previous release of *RTI Data Distribution Service* that is accessing those fields, data-type changes may be necessary.

1. *RTI Connext* .NET language binding is currently supported for C# and C++/CLI.

❑ **Changes in RtpsReliableWriterProtocol_t**

Starting with *RTI Data Distribution Service* 4.4c (and thereby included in *Connext*), two fields in DDS_RtpsReliableWriterProtocol_t have been renamed:

- Old name: disable_positive_acks_decrease_sample_keep_duration_**scaler**
  New name:disable_positive_acks_decrease_sample_keep_duration_**factor**
- Old name: disable_positive_acks_increase_sample_keep_duration_**scaler**
  New name: disable_positive_acks_increase_sample_keep_duration_**factor**

In releases prior to 4.4c, the NACK-only feature was not supported on platforms without floating-point support. Older versions of *RTI Data Distribution Service* will not run on these platforms because floats and doubles are used in the implementation of the NACK-only feature. In releases 4.4c and above, the NACK-only feature uses fixed-point arithmetic and the new DDS_Long "factor" fields noted above, which replace the DDS_Double "scaler" fields.

❑ **Tolerance for Destination-Ordering by Source-Timestamp**

Starting with *RTI Data Distribution Service* 4.4b (and thereby included in *Connext*), by default, the middleware is less restrictive (compared to older releases) on the writer side with regards to timestamps between consecutive samples: if the timestamp of the current sample is less than the timestamp of the previous sample by a small tolerance amount, **write()** will succeed.

If you are upgrading from *RTI Data Distribution Service* 4.4a or lower, and the application you are upgrading relied on the middleware to reject timestamps that 'went backwards' on the writer side (that is, when a sample's timestamp was earlier than the previous sample's), there are two ways to keep the previous, more restrictive behavior:

- If your DestinationOrderQosPolicy's **kind** is BY_SOURCE_TIMESTAMP: set the new field in the DestinationOrderQosPolicy, **source_timestamp_tolerance**, to 0.
- If your DestinationOrderQosPolicy's **kind** is BY_RECEPTION_TIMESTAMP on the writer side, consider changing it to BY_SOURCE_TIMESTAMP instead and setting **source_timestamp_tolerance** to 0. However, this may not be desirable if you had a particular reason for using BY_RECEPTION_TIMESTAMP (perhaps because you did not want to match readers with BY_SOURCE_TIMESTAMP). If you need to keep the BY_RECEPTION_TIMESTAMP setting, there is no QoS setting that will give you the exact same behavior on the writer side as the previous release.

Starting with *RTI Data Distribution Service* 4.4b (and thereby included in *Connext*), by default, the middleware is more restrictive (compared to older releases) on the reader side with regards to source and reception timestamps of a sample if DestinationOrderQosPolicy **kind** is set to BY_SOURCE_TIMESTAMP: if the reception timestamp of the sample is less than the source timestamp by more than the tolerance amount, the sample will be rejected.

If you are upgrading from *RTI Data Distribution Service* 4.4a or lower, your reader is using BY_SOURCE_TIMESTAMP, and you need the previous less restrictive behavior, set **source_timestamp_tolerance** to infinite on the reader side.

❑ **New Location and Name for Default XML QoS Profiles File (formerly NDDS_QOS_PROFILES.xml)**

Starting with *RTI Data Distribution Service* 4.4d (and thereby included in *Connext*) the default XML QoS Profiles file has been renamed and is installed in a new directory:

- Old location/name: **$NDDSHOME/resource/xml/NDDS_QOS_PROFILES.xml**
- New location/name: **$NDDSHOME/resource/qos_profiles_<*version*>/xml/NDDS_QOS_PROFILES.example.xml** (where <*version*> can be 4.4d, for example)

If you want to use this QoS profile, you need to set up your **NDDSHOME** environment variable at run time and rename the file **NDDS_QOS_PROFILES.example.xml** to **NDDS_QOS_PROFILES.xml** (i.e., by default, even if your **NDDSHOME** environment variable is set, this QoS profile is not used.) See Section 15.2 in the *RTI Core Libraries and Utilities User's Manual* for details.

❑ **Changes in the default value for the max_objects_per_thread field**

Starting with *RTI Data Distribution Service* 4.4d (and thereby included in *Connext*), the default value for the **max_objects_per_thread** field in the SystemResourceLimitsQosPolicy has been changed from 512 to 1024.

❑ **Type Change in Constructor for SampleInfoSeq—.NET[1] Only**

Starting with *RTI Data Distribution Service* 4.5c (and thereby included in *Connext*), the constructor for SampleInfoSeq has been changed from SampleInfoSeq(UInt32 maxSamples) to SampleInfoSeq(Int32 maxSamples). This was to make it consistent with other sequences.

❑ **Default Send Window Sizes Changed to Infinite**

- Starting with *RTI Data Distribution Service* 4.5d (and thereby included in *Connext*), the send window size of a *DataWriter* is set to infinite by default. This is done by changing the default values of two fields in DDS_RtpsReliableWriterProtocol_t (**min_send_window_size, max_send_window_size**) to be DDS_LENGTH_UNLIMITED.
- In *RTI Data Distribution Service* 4.4d, the send window feature was introduced and was enabled by default in 4.5c (with **min_send_window_size** = 32, **max_send_window_size** = 256). For *DataWriters* with a HistoryQosPolicy **kind** of KEEP_LAST, enabling the send window could cause writes to block, and possibly fail due to blocking timeout. This blocking behavior changed the expected behavior of applications using default QoS. To preserve that preestablished non-blocking default behavior, the send window size has been changed to be infinite by default starting in release 4.5d.
- Users wanting the performance benefits of a finite send window will now have to configure the send window explicitly.

2.3 Extensible Types Compatibility

Connext 5.0 includes partial support for the "Extensible and Dynamic Topic Types for DDS" (DDS-XTypes) specification[2] from the Object Management Group (OMG). This support allows systems to define data types in a more flexible way, and to evolve data types over time without giving up portability, interoperability, or the expressiveness of the DDS type system.

For information related to compatibility issues associated with the Extensible Types support, see the new *Core Libraries and Utilities Getting Started Guide Addendum for Extensible Types* (**RTI_CoreLibrariesAndUtilities_GettingStarted_ExtensibleTypesAddendum.pdf**).

1. *RTI Connext* .NET language binding is currently supported for C# and C++/CLI.
2. http://www.omg.org/spec/DDS-XTypes/

2.4 ODBC Database Compatibility

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

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

We have tested the following driver:

- MySQL ODBC 5.1.44

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

To use MySQL, you will also need the MySQL ODBC 5.1.6 (or higher) driver. For non-Windows platforms, UnixODBC 2.2.12 (or higher) is also required.

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

- **AIX:** p5AIX5.3xlc9.0, 64p5AIX5.3xlc9.0
- **Linux:** i86Linux2.6gcc3.4.3, x64Linux2.6gcc3.4.5; i86Linux2.6gcc4.1.1, x64Linux2.6gcc4.1.1; i86Linux2.6gcc4.4.3, x64Linux2.6gcc4.4.3
- **Solaris:** sparcSol2.10gcc3.4.2, sparc64Sol2.10gcc3.4.2
- **Windows:** i86Win32VS2005, i86Win32VS2008, i86Win32VS2010, x64Win64VS2010

For more information on database setup, please see the *Addendum for Database Setup* (**RTI_CoreLibrariesAndUtilities_GettingStarted_DatabaseAddendum.pdf**).

2.5 Transport Compatibility

The shared memory transport in *Connext* does not interoperate with the shared memory transport in previous releases of *RTI Data Distribution Service*.

If two applications, one using *Connext* and one using *RTI Data Distribution Service*, run on the same node and they have the shared memory transport enabled, they will fail with the following error:

```
[D0004|CREATE Participant|D0004|ENABLE]
NDDS_Transport_Shmem_is_segment_compatible:incompatible shared
memory protocol detected.
Current version 1.0 not compatible with 2.0.
```

A possible workaround for this interoperability issue is to disable the shared memory transport and use local communications over UDPv4 by setting **participant_qos.transport_builtin** to DDS_TRANSPORTBUILTIN_UDPv4.

If you have an interoperability requirement and you cannot switch to UDPv4, please contact **support@rti.com**.

3 What's Fixed in Connext 5.0.0

This section describes bugs fixed since the release of *Connext* 4.5f.

3.1 Fixes Related to Batching

3.1.1 Unkeyed DataReader using KEEP_LAST History Sometimes Stopped Receiving Data

This is a rare situation that only occurs when the DataReader receives samples from a DataWriter using batching and when the DataReader have loans on samples that are replaced after applying the history depth.

[RTI Issue ID CORE-5097, Bug # 14324]

3.2 Fixes Related to Content Filtering

3.2.1 get_matched_subscription_data() Returned Empty Structure for content_filter_property

The *DataWriter's* **get_matched_subscription_data()** operation populates a DDS_SubscriptionBuiltinTopicData structure. However the **content_filter_property** member of that structure contained all NULLs, even if there was a *DataReader* on a matched ContentFilteredTopic. This problem has been resolved.

[RTI Issue ID CORE-5079]

3.2.2 Incorrect Filtering Applied to DataReaders with Non-Volatile Durability if DataWriter used ON_DEMAND Refiltering

Content-filtering for *DataReaders* with non-volatile Durability may not have worked as expected in some cases. The *DataReader* may have either received samples that should have been filtered out, or may have *not* received samples it should have gotten. This problem only existed if the *DataWriter's* HistoryQosPolicy **refilter** field was set to ON_DEMAND.

[RTI Issue ID CORE-4131]

3.2.3 Parsing Error if Multi-Channel DataWriter had Empty Filter Expression

If the filter expression for a channel of a multi-channel *DataWriter* was set to the empty string (for example, by using <filter_expression></filter_expression> in the XML configuration file), you may have seen a parser error. This occurred if a *DataReader* was discovered that used a ContentFilteredTopic based on the same topic used by the multi-channel *DataWriter*. This problem had been resolved; now an empty filter expression (empty string) is allowed for multi-channel *DataWriters*.

[RTI Issue ID CORE-5431]

3.2.4 SqlFilter Compile Error with Enumeration Values—Java API Only

In previous releases, after a *DomainParticipant* discovered a *DataReader* with a ContentFilteredTopic, the compilation of the *DataReader's* filter expression may have failed with the following error if the filter expression contained enumeration values:

```
SqlFilter.compile:Combination of different enum types not allowed.
ContentFilteredTopicImpl_forward_compile:!compile expression
```

This problem had been resolved.

[RTI Issue ID CORE-4200]

3.3 Fixes Related to Reliability Protocol

3.3.1 DataReader Processed Duplicate Heartbeat Messages

A *DataReader* may have processed duplicate heartbeat messages (caused by the presence of redundant communication paths) multiple times. This problem has been resolved.

[RTI Issue ID CORE-5136]

3.3.2 DataWriter Protocol Statistics did not Include First HeartBeat Sent to VOLATILE DataReader

Statistics retrieved from the following *DataWriter* operations may not have accounted for the first heartbeat (HB) message sent to a *DataReader* configured with VOLATILE Durability.

- **get_datawriter_protocol_status()**
- **get_matched_subscription_datawriter_protocol_status()**
- **get_matched_subscription_datawriter_protocol_status_by_locator()**

This problem has been resolved.

[RTI Issue ID CORE-5149]

3.3.3 Reliable DataWriter Sending Large Data may not have Inactivated Non-Progressing DataReader

When **DDS_RtpsReliableWriterProtocol_t.inactivate_nonprogressing_readers** is set to TRUE, a reliable *DataWriter* should treat any *DataReader* that is not showing progress in receiving and acknowledging samples (after **max_heartbeat_retries**) as inactive. However, for reliable *DataWriters* sending large samples that are fragmented by the middleware, a problem existed where a *DataWriter* did not inactivate a *DataReader* that was continuously requesting for the same data fragment to be resent. This problem has been resolved.

[RTI Issue ID CORE-5163]

3.3.4 Unexpected Timeout Error from Write Operation

This problem only occurred in version 4.5e Rev. 07 when a *DataWriter* was configured for strict reliability (that is, Reliability QoS **kind** = Reliable and History QoS **kind** = KEEP_ALL) and its ResourceLimits QoS **max_samples** was set to a finite value (which is not the default case).

Even if the existing reliable *DataReaders* had acknowledged all the samples in the *DataWriter's* queue, the write operation may have returned DDS_RETCODE_TIMEOUT when the queue filled up. The write operation blocked until the next periodic HeartBeat message was sent. This situation only occurred when a *DataReader* was matched.

[RTI Issue ID CORE-5214]

3.3.5 False Reporting of Samples Lost if Late-Joining Readers did not Include Expired Samples

DataWriters whose LifespanQosPolicy was set to a finite value did not deliver samples with expired lifespan to *DataReaders*. However, in the case of late-joining *DataReaders*, the initial HeartBeat messages from the *DataWriter* to the *DataReader* did include the expired samples. Subsequently, the *DataReader* would request these samples, but the *DataWriter* would not send them. This caused the *DataReader* to report samples lost to the application.

This behavior has been changed so that the HeartBeat message now includes samples with expired lifespans, thus removing the spurious samples lost on the reader side.

[RTI Issue ID CORE-5074; Bug # 14374]

3.3.6 Lowered Verbosity for "ACK Ignored" Messages

If a *DataWriter* dropped an ACKNACK message, *Connext* would log a warning ("ACK ignored ..."). However, an ACK might also be ignored because of a valid scenario (described below);

therefore the verbosity for the "ACK ignored" message has been changed from WARNING to REMOTE.

For example, this is a valid scenario in which an ACK may be ignored: A *DataWriter* has just inactivated a remote *DataReader* and the ACKNACK is stale (sent before the *DataWriter* inactivated the *DataReader*). This situation is a false-alarm.

In this release, the verbosity of the "ACK ignored" messages has been changed to REMOTE instead of WARNING.

[RTI Issue ID CORE-5170]

3.4 Fixes Related to Transports

3.4.1 Possible Errors when Using Shared Memory on PPC SMP Board with VxWorks

When using the shared memory transport on a PPC SMP board running VxWorks, you may have seen the following error messages:

```
PRESCstReaderCollator_storeInlineQos:!serialized sample to keyhash
PRESCstReaderCollator_newAnonData:!add store inline QoS
```

This problem has been resolved.

[RTI Issue ID CORE-5122]

3.4.2 Shared Memory Send Failures not Reported

In previous *Connext* releases, the shared memory transport did not print any messages when the execution of the send operation failed because the shared memory transport queue was full. This release prints a message indicating this situation with a NDDS_CONFIG_LOG_VERBOSITY_STATUS_LOCAL verbosity level.

[RTI Issue ID CORE-3457]

3.5 Fixes Related to Dynamic Data

3.5.1 Corrected Implementation of DynamicData's get/set_wstring API

In the previous release, Connext did not convert the wchar_t representation (2 bytes) to the DDS_Wchar representation (4 bytes). This caused an incompatible representation. For example, when sending a wstring containing "Hello World" the receiving application may have received the wrong content (such as just the "H".) This problem has been resolved.

[RTI Issue ID CORE-4204, Bug # 14316]

3.5.2 Issues when Using Untyped Methods on Dynamic DataWriter—Java API Only

Some applications may have generic, type-independent modules. Their code uses the com.rti.dds.publication.DataWriter interface and its untyped methods (**write_untyped(), dispose_untyped()**, **register_instance_untyped()** and **unregister_instance_untyped()**).

When the actual object was a DynamicDataWriter, using those methods resulted in undefined behavior. This problem has been resolved; using the untyped methods is now equivalent to using the type-specific ones. For example, the two write operations below are equivalent:

```
DynamicDataWriter dynamicDataWriter = publisher.create_datawriter(...);
DataWriter writer = dynamicDataWriter;
DynamicData sample = typeSupport.create_data();
dynamicDataWriter.write(sample, InstanceHandle_t.HANDLE_NIL);
writer.write_untyped(sample, InstanceHandle_t.HANDLE_NIL);
```

[RTI Issue ID CORE-5215]

3.5.3 Creating Two ContentFilteredTopics Based on DynamicData Topic on Same Participant Failed—Java API Only

After creating a ContentFilteredTopic based on a DynamicData topic, an attempt to create another ContentFilteredTopic, based on the same or a different DynamicData topic, on the same Participant would fail. This problem has been resolved; now any number of ContentFilteredTopics can be created. This problem only occurred when using the Java API.

[RTI Issue ID CORE-4227, Bug # 14354]

3.5.4 Possible Deserialization Error When Using Dynamic Data and Extensible Types

There was a possible deserialization error when using Dynamic Data and Extensible Types. If a *DataReader* expected an Extensible type, but received a Base type instead, you may have seen this error:

```
DDS_DynamicData_initialize_from_bufferI:ERROR: Failed to get sample init
error
```

This issue has been resolved.

[RTI Issue ID CORE-5340]

3.6 Fixes Related to XML Parser

3.6.1 Calls to create_<entity>_with_profile would Hang if DomainParticipant used Monitoring Library

If the *DomainParticipant* was configured to use *RTI Monitoring Library*, calls to the following operations would hang:

- *Publisher's* **create_datawriter_with_profile()**
- *Subscriber's* **create_datareader_with_profile()**
- *DomainParticipant's* **create_topic_with_profile()**

This problem has been resolved.

[RTI Issue ID CORE-5152]

3.6.2 No Error Reported when QoS Profile Tag Defined Multiple Times in XML QoS Profile

The XML configuration parser failed to report an error if a QoS profile tag was defined multiple times within an XML QoS Profile. For example:

```
<qos_library name="PropertyInheritanceLib">
  <qos_profile name="BaseProfile">
    <datareader_qos>
      <property>
          ...
      </property>

      <property>
          ...
      </property>
    </datareader_qos>
  </qos_profile>
</qos_library>
```

This problem has been resolved. Now this erroneous configuration will be reported as an error.

[RTI Issue ID CORE-5204]

3.7 Fixed Related to Read Conditions

3.7.1 ReadCondition Reported Erroneous Status for Keyed Reader Following NOT_ALIVE Status

For keyed types, if an instance became NOT_ALIVE because it was disposed or due to a NO_WRITERS condition, if and when the instance returned to ALIVE status, ReadConditions for that *DataReader* misreported available samples when they were not present. Since the sample was not really present, it could not be taken; therefore the condition could not be cleared. This problem has been resolved.

[RTI Issue ID CORE-5178]

3.7.2 ReadCondition Created from Disabled Reader Never Triggered

If a ReadCondition was created while its associated *DataReader* was disabled, the ReadCondition was never triggered (even if the *DataReader* was subsequently enabled). ReadConditions that were created (or re-created) after the *DataReader* was enabled were unaffected. This problem has been resolved; now a ReadCondition can be created while a *DataReader* is disabled and it will function correctly once the *DataReader* is enabled.

[RTI Issue ID CORE-5324]

3.8 Fixes Related to Experimental Features

3.8.1 Incomplete Cleanup if create_participant_from_config_exp() Failed to Create Underlying Entities

A call to **create_participant_from_config_exp()** creates a *DomainParticipant* and all the underlying entities defined in the corresponding configuration. If an error occurs while creating one of these entities (after the *DomainParticipant* has been created), **create_participant_from_config()** *should* delete all the entities that it created before encountering the failure, then return NULL. However, the *DomainParticipant* and any underlying entities that were created were not deleted before returning. This resulted in an error if **finalize()** was invoked on the DomainParticipant-Factory. This problem has been resolved.

[RTI Issue ID CORE-5071]

3.9 Other Fixes

3.9.1 Potential Precondition Errors if Asynchronous Publisher Feature Used by Persistence Service DataWriter or DataWriter Using Durable Writer History

A *DataReader* receiving fragmented samples from a *DataWriter* configured to use durable writer history or from a *Persistence Service* DataWriter configured in PERSISTENT mode may report precondition errors that will cause samples to be dropped.

This problem has been fixed.

[RTI Issue ID CORE-5345]

3.9.2 Potential Segmentation Fault on QNX Architectures

Previous *Connext* releases may have issued a segmentation fault on QNX architectures due to the internal usage of **random()**, which is a non-reentrant function. This problem has been resolved by replacing **random()** with **rand_r()**.

[RTI Issue ID CORE-5294]

3.9.3 DDS Built-in Types did not Include C++ Traits

In release 4.5f, types generated with *rtiddsgen* for C++ included traits that allowed the type of related DDS entities (such us *DataWriter*, *DataReader*, Sequence types, etc.) to be inferred.

This information was not available for the DDS built-in types (Octets, KeyedOctets, String, and KeyedString). Therefore the templatized functions that used these traits could not be instantiated with one of the built-in types.

This problem has been resolved for Octets, KeyedOctets and KeyedString. The String built-in type is a special case (char* is a primitive type) and this feature is not currently supported.

[RTI Issue ID CORE-5096]

3.9.4 Some DDS Infrastructure Types Lacked Copy Constructors—C++ API Only

The C++ API for some infrastructure types lacked copy constructors. This affected QoS policies and the built-in types DDS_KeyedString, DDS_Octets and DDS_KeyedOctets. This problem made it impossible to make copies or pass parameters by value. This problem has been resolved and copy constructors have been added.

[RTI Issue ID CORE-2588; Bug # 12181]

3.9.5 DataReaders with Default Character Encoding Other Than ISO-8859-1 may have Received Corrupt Characters when using IDL Types with Strings—Java API Only

DataReaders that used default character encoding other than ISO-8859-1 could have received corrupt characters if the IDL type included strings. This did not affect common English characters, but may have been an issue with special characters in other languages, such as German characters ö, ä, and ü. This problem has been resolved.

[RTI Issue ID CORE-3825; Bug # 13793]

3.9.6 Calls to wait_for_historical_data() Could Return Prematurely

If **wait_for_historical_data()** was called while there were multiple remote *DataWriters* matched with the calling *DataReader*, it was possible for **wait_for_historical_data()** to return before all of the historical data had been received. This problem has been resolved.

[RTI Issue ID CORE-5076; Bug # 14361]

3.9.7 Participant Creation Failure if Network Interface had IP Address 0.0.0.0

DomainParticipant creation may have failed with the following errors if a network interface had the invalid IP address 0.0.0.0:

```
[CREATE Participant]COMMENDActiveFacade_new:!precondition
[CREATE Participant]PRESParticipant_new:!create facade
[CREATE Participant]DDS_DomainParticipantPresentation_initialize:
!create participant core
[CREATE Participant]DDS_DomainParticipant_createI:!create presentation participant
[CREATE Participant]DDS_DomainParticipantFactory_create_participant_disabledI:
!create participant
DDSDomainParticipant_impl::create_disabledI:!create participant
DDSDomainParticipant_impl::createI:!create participant
DomainParticipantFactory_impl::create_participant():!create failure creating
participant create_participant error
```

This problem has been resolved.

[RTI Issue ID CORE-5103]

3.9.8 Unreachable Address not Included in Error Message

During the discovery process, participants receive other participants' locator lists. If the locator list contained an unreachable address, the resulting error message did not include the address. For example, you may have seen this message:

```
RTINetioSender_addDestination: no transport for request
```

In this release, the error message includes the address that could not be reached.

For example, if no transport can be found for a UDPv4 address, the message will resemble:

```
RTINetioSender_addDestination:no transport for destination request udpv4://
10.30.1.205:21410
```

If no transport can be found for a SHMEM address, you will see a message such as:

```
RTINetioSender_addDestination:no transport for destination request shmem://
0002:0007:0105:0000:0000:0000:0000:0000:21410
```

For other transports, the message will be:

```
<transport class name>://xxxx:xxxx:xxxx:xxxx:xxxx:xxxx:xxxx:<port>
```

{RTI Issue ID CORE-5220]

3.9.9 VxWorks 6.7 RTP Mode Support for SMP Boards

The *Connext* libraries for the VxWorks 6.7 RTP mode platform used deprecated VxWorks system calls, which made the libraries incompatible on multi-processor boards. The deprecated calls are no longer used and VxWorks 6.7 RTP libraries can now be used on single- and multi-processor boards.

[RTI Issue ID CORE-5265]

3.9.10 Possible Segmentation Fault or Precondition Error if Cumulative String Length of Properties Exceeded Maximum

If the **participant_property_string_max_length** in the DomainParticipantResourceLimitsQosPolicy was greater than the *DomainParticipant's* properties' cumulative length *without* the null terminating characters, but less than or equal to the cumulative length *with* the null terminating characters, you may have seen either a segmentation fault when using the release libraries, or the following errors when using the debug libraries:

```
[D0000|CREATE Participant|D0000|ENABLE]PRESPropertyQosPolicy_copy:
 !precondition
[D0000|CREATE Participant|D0000|ENABLE]PRESParticipant_getProperty:
 !copyPropertyPolicy
[D0000|CREATE Participant|D0000|ENABLE]DDS_DomainParticipantPresentation
 _get_qos:ERROR: Failed to get QoS
[D0000|CREATE Participant|D0000|ENABLE]DDS_DomainParticipant_enableI:
 ERROR: Failed to get participant QoS
[D0000|CREATE Participant]DDS_DomainParticipantFactory_create_participant:
 ERROR: Failed to auto-enable entity
```

For example, this problem would occur if **participant_property_string_max_length** was 10 and there were two properties: <name="12", value="34"> and <name="567", value="890">. Since the cumulative length with the null terminating characters was 14, which is greater than 10, the correct behavior is for *Connext* to print these error messages:

```
[CREATE Participant]DDS_ResourceLimitsQosPolicy_is_consistentI:inconsistent
 QoS policies: cumulative string length of the properties and
 DDS_DomainParticipantQos.resource_limits.participant_property_string_
 max_length
[CREATE Participant]DDS_DomainParticipantQos_is_consistentI:inconsistent QoS
 policy: property
[CREATE Participant]DDS_DomainParticipantFactory_set_default_participant_qos:
 ERROR:Inconsistent QoS
```

[RTI Issue ID CORE-5288]

3.9.11 Reader Memory Growth if Number of Received Instances Exceeded max_total_instances

If a *DataReader* received more instances than its **reader_resource_limits.max_total_instances**, *Connext* did not completely free the memory associated with the replaced existing instance. This

led to memory growth whenever an existing instance was replaced. This problem has been resolved.

[RTI Issue ID CORE-5302]

3.9.12 Possible Incorrect Instance Handle Received if disable_inline_keyhash Enabled—.NET APIs only

If a *DataWriter* set the DataWriterProtocolQosPolicy's **disable_inline_keyhash** to TRUE, a subscribing application with multiple matching *DataReaders* for that *DataWriter* may have received an incorrect instance handle. A *DataReader* may have retrieved the instance handle for a sample that was received by a different *DataReader*. This could have caused the *DataReader* to drop a sample or get incorrect meta-data for a sample. This problem, which only affected the C# and C++/CLI APIs, has been resolved.

[RTI Issue ID CORE-5303]

3.9.13 Small channel_seq_max_length Resulted in Segmentation Fault

A small value for the DomainParticipantResourceLimitsQosPolicy's **channel_seq_max_length** (between 1 and 4, inclusive) may have resulted in a segmentation fault. This problem has been resolved.

[RTI Issue ID CORE-5332]

3.9.14 Errors from RTI Spy and Ping Tools when Using VxWorks Kernel Mode

The *rtiddsspy* and *rtiddsping* tools could not be used on VxWorks platforms using kernel mode. You may have seen "undefined reference" errors. This problem has been resolved.

[RTI Issue ID CORE-5353]

4 Known Issues

4.1 AppAck Messages Cannot be Greater Than Underlying Transport Message Size

A *DataReader* with **acknowledgment_kind** (in the ReliabilityQosPolicy) set to DDS_APPLICATION_AUTO_ACKNOWLEDGMENT_MODE or DDS_APPLICATION_EXPLICIT_ACKNOWLEDGMENT_MODE cannot send AppAck messages greater than the underlying transport message size.

If a *DataReader* tries to send an AppAck message greater than the transport message size, *Connext* will print the following error message:

```
COMMENDFacade_sendAppAck:!add APP_ACK to MIG
COMMENDSrReaderService_sendAppAck:!send APP_ACK
PRESPsService_onReaderAppAckSendEvent:!send acknowledgment
```

To recover from the above error, the *DataReader* must acknowledge samples until the size of the AppAck message goes below the transport message size threshold.

Why does an AppAck message increase its size?

An AppAck message contains a list of sequence number intervals where each interval represents a set of consecutive sequence numbers that have been already acknowledged.

As long as samples are acknowledged in order, the AppAck message will always have a single interval. However, when samples are acknowledged out of order, the number of intervals and the size of the AppAck will increase.

For additional information, see Section 6.3.12, *Application Acknowledgment*, in the *RTI Core Libraries and Utilities User's Manual.*

[RTI Issue ID CORE-5329]

4.2 DataReader Cannot Persist AppAck Messages Greater Than 32767 Bytes

A *DataReader* using durable reader state, whose **acknowledgment_kind** (in the ReliabilityQosPolicy) is set to DDS_APPLICATION_AUTO_ACKNOWLEDGMENT_MODE or DDS_APPLICATION_EXPLICIT_ACKNOWLEDGMENT_MODE, cannot persist an AppAck message greater than 32767 bytes.

To recover from the previous error, the *DataReader* must acknowledge samples until the size of the AppAck message goes below the transport message size threshold.

For additional information, see Section 12.4, *Durable Reader State*, in the *RTI Core Libraries and Utilities User's Manual*.

[RTI Issue ID CORE-5360]

4.3 Request and Reply Topics Must be Created with Types Generated by rtiddsgen—C API Only

When using the C API to create Request and Reply Topics, these topics must use data types that have been generated by *rtiddsgen*. Other APIs support using built-in types and DynamicData types.

[RTI Issue ID BIGPINE-537]

4.4 Writer-Side Filtering May Cause Missed Deadline

If you are using a ContentFilteredTopic and you set the Deadline QosPolicy, the deadline may be missed due to filtering by a *DataWriter*.

[RTI Issue ID CORE-1634, Bug # 10765]

4.5 Writer-side Filtering Functions Can be Invoked Even After Filter Unregistered

If you install a ContentFilter that implements the writer-side filtering APIs, *Connext* can call those APIs even after the ContentFilter has been unregistered.

[RTI Issue ID CORE-5356]

4.6 Disabled Interfaces on Windows Systems

The creation of a *DomainParticipant* will fail if no interface is enabled *and* the **DiscoveryQosPolicy.multicast_receive_addresses** list (specified either programmatically, or through the **NDDS_DISCOVERY_PEERS** file or environment variable) contains a multicast address.

However, if **NDDS_DISCOVERY_PEERS** only contains unicast addresses, the *DomainParticipant* will be successfully created even if all the interfaces are disabled. The creation of a *DataReader* will fail if its TransportMulticastQosPolicy contains a UDPv4 or UPDv6 multicast address.

4.7 Wrong Error Code After Timeout on write() from Asynchronous Publisher

When using an asynchronous publisher, if **write()** times out, it will mistakenly return DDS_RETCODE_ERROR instead of the correct code, DDS_RETCODE_TIMEOUT.

[RTI Issue ID CORE-2016, Bug # 11362]

4.8 Incorrect Content Filtering for Valuetypes and Sparse Types

Content filters may not filter correctly if (a) the type is a valuetype or sparse type using inheritance and (b) the filters refer to members of a derived class.

This issue exists for Topics using DynamicData type support. It may also affect filtering of valuetypes using the .NET[1] or C APIs, or CORBA-compatible C++ type plugins.

[RTI Issue ID CORE-2949, Bug # 12606]

4.9 Code Generation for Inline Nested Structures, Unions, and Valuetypes not Supported

Code generation for inline nested structures, unions, and valuetypes is not supported. For example, *rtiddsgen* will produce erroneous code for these structures:

IDL:

```
struct Outer {
    short outer_short;
    struct Inner {
        char inner_char;
        short inner_short;
    } outer_nested_inner;
};
```

XML:

```
<struct name="Outer">
    <member name="outer_short" type="short"/>
    <struct name="Inner">
        <member name="inner_char" type="char"/>
        <member name="inner_short" type="short"/>
    </struct>
</struct>
```

[RTI Issue ID CODEGEN-54, Bug # 9014]

4.10 .NET Code Generation for Multi-dimensional Arrays of Sequences not Supported

The .NET code generated by *rtiddsgen* for multi-dimensional arrays of sequences is not correct and will not compile.

For example:

```
struct MyStruct {
    sequence<short, 4> m1[3][2];
};
```

[RTI Issue ID CODEGEN-376, Bug # 13088]

4.11 Memory Leak in Applications using TCP Transport in Asymmetric Mode

If an application uses the TCP transport in asymmetric mode (server_bind_port = 0), a memory leak may occur. The size of the memory leak depends on the number of TCP connections opened by the transport and the value of the transport property **parent.message_size_max**.

[RTI Issue ID COREPLG-63, Bug # 14313]

1. *RTI Connext* .NET language binding is currently supported for C# and C++/CLI.

4.12 Issues with Dynamic Data

- The conversion of data by member-access primitives (**get_X()** operations) is limited when converting to types that are not supported on all platforms. This limitation applies when converting to a 64-bit long long type (**get_longlong()** and **get_ulonglong()** operations) and a 128-bit long double type (**get_longdouble()**). These methods will always work for data members that are actually of the correct type, but will only support conversion from values that are stored as smaller types on a subset of platforms. Conversion to 64-bit long longs from a 32-bit or smaller integer type is supported on all Windows, Solaris, and Linux architectures, and any additional 64-bit architectures. Conversion to 128-bit long doubles from a float or double is only supported on Solaris SPARC architectures.

  [RTI Issue ID CORE-2986, Bug # 12647]

- DynamicData cannot handle a union with a discriminator that is set to a value which is not defined in the type.

  [RTI Issue ID CORE-3142, Bug # 12855]

- DynamicData may have problems resizing variable-size members that are >= 64k in size. In this case, the method (**set_X()** or **unbind_complex_member()**) will fail with the error: "sparsely stored member exceeds 65535 bytes." Note that it is not possible for a member of a sparse type to be >= 64k.

  [RTI Issue ID CORE-3177, Bug # 12897]

- Topics of DynamicData types that contain bit fields are not supported by *rtiddsspy*.

  [RTI Issue ID CORE-3949, Bug # 13949]

- DynamicData does not support out-of-order assignment of members that are longer than 65,535 bytes. In this situation, the DynamicData API will report the following error:

  ```
  sparsely stored member exceeds 65535 bytes
  ```

  For example:

  ```
  struct MyStruct {
         string<131072> m1;
  string<131072> m2;
  };
  ```

  With the above type, the following sequence of operations will fail because m2 is assigned before m1 and has a length greater than 65,535 characters.

  ```
  str = DDS_String_alloc(131072);
  memset(str, 'x', 131072);
  str[131071]= 0;
  DDS_DynamicData_set_string(
    data, "m2",  DDS_DYNAMIC_DATA_MEMBER_ID_UNSPECIFIED, str);
  DDS_DynamicData_set_string(
     data, "m1", DDS_DYNAMIC_DATA_MEMBER_ID_UNSPECIFIED, str);
  ```

  If member m1 is assigned *before* m2, the sequence of operations will succeed.

  [RTI Issue ID CORE-3791, Bug # 13745]

5 Custom Supported Platforms

Table 5.1 lists additional target libraries available with *Connext*, 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 5.1 **Custom Supported Platforms**

Operating System		CPU	Compiler	RTI Architecture Abbreviation
INTEGRITY	INTEGRITY 5.0.11	PPC8349	GHnet2 TCP/IP stack	ppc8349Inty5.0.11.mds8349
Linux	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
	Red Hat Enterprise Linux 5.2 (2.6 kernel)	Pentium class	gcc 4.2.1	i86Linux2.6gcc4.2.1
			Sun Java Platform Standard Edition JDK 1.6	i86Linux2.6gcc4.2.1jdk
	Red Hat Enterprise Linux 6 for IBM POWER7 Servers (2.6.32-70.el.ppc64)	POWER7	gcc 4.4.4	power7Linux2.6gcc4.4.4
	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
	RedHawk Linux 5.4 (2.6 kernel)	Pentium class	gcc 4.2.1	i86RedHawk5.4gcc4.2.1
			Sun Java Platform Standard Edition JDK 1.6	i86RedHawk5.4gcc4.2.1jdk
	RedHawk Linux 6.0	x64	gcc 4.4.5	x64Linux2.6gcc4.4.5
	Wind River Linux 3.0.3 (2.6 kernel) Note: This platform is only available as a Custom Target Library (CTL). See Section 5.	Pentium class	gcc 4.3.2	i86WRLinux2.6gcc4.3.2

6 Experimental Features

Experimental features are used to evaluate potential new features and obtain customer feedback. They are not guaranteed to be consistent or supported and they should not be used in production.

The APIs for experimental features use the suffix **_exp** to distinguish them from other APIs. For example:

```
const DDS::TypeCode * DDS_DomainParticipant::get_typecode_exp(
    const char * type_name);
```

In the API Reference Documentation[1], experimental APIs are marked with **<<experimental>>**.

Experimental features are clearly documented as such in the *Core Libraries and Utilities What's New* document or the *Release Notes* document of the component in which they are included, as well as in the component's *User's Manual*.

Disclaimers

- ❏ The experimental feature APIs may be only available in a subset of the supported languages and for a subset of the supported platforms.
- ❏ The names of experimental feature APIs will change if they become officially supported. At the very least, the suffix, **_exp**, will be removed.
- ❏ Experimental features may or may not appear in future product releases.
- ❏ Experimental features should not be used in production.

Please submit your comments and suggestions about experimental features to **support@rti.com** or via the RTI Customer Portal (https://support.rti.com/).

1. API reference documentation is provided in both HTML and PDF formats for all supported languages.