RTI Connext DDS

Core Libraries and Utilities

Release Notes

Version 5.1.0

Your systems. Working as one.

Printed in U.S.A. First printing.
December 2013.

Trademarks

Copy and Use Restrictions

Technical Support

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

Contents

Release Notes

This document includes the following sections:

- ❏ System Requirements (Section 1)
- ❏ Compatibility (Section 2)
- ❏ What's Fixed in 5.1.0 (Section 3)
- ❏ Known Issues (Section 4)
- ❏ Experimental Features (Section 5)

For an overview of new features, please see the *What's New* document[1].

Many readers will also want to look at additional documentation available online. In particular, RTI recommends the following:

- ❏ Use the RTI Customer Portal (http://support.rti.com) to download RTI software, access documentation and contact RTI Support. The RTI Customer Portal requires a username and password. You will receive this in the email confirming your purchase. If you do not have this email, please contact **license@rti.com**. Resetting your login password can be done directly at the RTI Customer Portal.
- ❏ The RTI Community website (http://community.rti.com) provides a wealth of knowledge to help you use *RTI Connext DDS*, including:
  - Best Practices,
  - Example code for specific features, as well as more complete use-case examples,
  - Solutions to common questions,
  - A glossary,
  - Downloads of experimental software,
  - And more.
- ❏ Whitepapers and other articles are available from http://www.rti.com/resources.

1. RTI_CoreLibrariesAndUtilities_WhatsNew.pdf

1 System Requirements

1.1 Supported Operating Systems

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

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 1.1 lists the platforms available with *Connext* 5.1.0.

Table 1.1 **Platforms Available with Connext 5.1.0**

Platform	Operating System	Reference
AIX®	AIX 5.3, 7.1	Table 1.3 on page 6
INTEGRITY® (target only)	INTEGRITY 5.0.11 and 10.0.2	Table 1.4 on page 6
Linux® (ARM® CPU)	Raspbian Wheezy 7.0 (3.x kernel)	Table 1.5 on page 6
Linux (Cell BE™ CPU)	Fedora™ 12 (2.6.32 kernel)	Table 1.6 on page 6
Linux (Intel® CPU)	CentOS 5.4, 5.5, 6.0, 6.2 - 6.4 (2.6 kernel) Fedora 12 (2.6.32 kernel) Fedora 12 (2.6.32 kernel) with gcc 4.5.1 Red Hat® Enterprise Linux 4.0, 5.0-5.2, 5.4, 5.5, 6.0 - 6.4 (2.6 kernel) Red Hat Enterprise Linux 5.2 with Real-Time Extensions (2.6 kernel) SUSE® Linux Enterprise Server 11 SP2 (2.6 kernel) SUSE Linux Enterprise Server 11 SP2 (3.x kernel) Ubuntu® Server 12.04 LTS (3.x kernel) Wind River® Linux 4 (2.6 kernel)	Table 1.7 on page 6
Linux (PowerPC® CPU) (target only)	Freescale™ P2020RDB (2.6.32 kernel) SELinux (2.6.32 kernel) Wind River® Linux 3 Yellow Dog™ Linux 4.0	Table 1.8 on page 7
LynxOS® (target only)[1]	LynxOS 4.0, 4.2, 5.0	Table 1.9 on page 8
Mac OS®	Mac OS X 10.8	Table 1.10 on page 8
QNX® (target only)	QNX Neutrino® 6.4.1, 6.5	Table 1.11 on page 8
Solaris™	Solaris 2.9, 2.10	Table 1.12 on page 8

Table 1.1 **Platforms Available with Connext 5.1.0**

Platform	Operating System	Reference
VxWorks® (target only)	VxWorks 5.5, 6.3 - 6.9	Table 1.13 on page 8
	VxWorks 653 2.3	
	VxWorks MILS 2.1.1	
Windows®	Windows 7 (32-bit and 64-bit Editions)	Table 1.14 on page 10
	Windows 8 (32-bit and 64-bit Editions)	
	Windows Server 2003 (32-bit and 64-bit Editions)	
	Windows Server 2008 R2 (64-bit Edition)	
	Windows Server 2012 R2 (64-bit Edition)	
	Windows Vista® (32-bit and 64-bit Editions)	
	Windows XP Professional SP 2(32-bit and 64-bit Editions)	

1. The Java API is not supported on LynxOS platforms in 5.1.0. If your application requires support for Java on LynxOS, please contact your RTI account manager.

Table 1.2 lists additional target libraries available for *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 1.2 **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	NI Linux Real-Time 3.2[1]	ARMv7	gcc 4.4.1	armv7AngstromLinux3.2 gcc4.4.1.cortex-a9
	Red Hat Enterprise Linux 5.2 (2.6 kernel)	x86	gcc 4.2.1	i86Linux2.6gcc4.2.1
			Java Platform, Standard Edition JDK 1.7	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 6.0	x64	gcc 4.4.5	x64Linux2.6gcc4.4.5
	Wind River Linux 3.0.3 (2.6 kernel)	x86	gcc 4.3.2	i86WRLinux2.6gcc4.3.2
VxWorks	VxWorks 6.7	Any PowerPC CPU with floating-point hardware that is backwards-compatible with 32-bit PowerPC 604[2]	JamaicaVM 6.2.1 with gcc 4.1.2	For Kernel Modules: ppc604Vx6.7gcc4.1.2jdk
	VxWorks 6.8			For Kernel Modules: ppc604Vx6.8gcc4.1.2jdk

1. Requires NI-RIO 13.1 release or a patch from NI for NI-RIO 13.0
2. Some PowerPC cores such as e500v1 and e500v2 are not fully backwards-compatible with PPC 604.

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

Visual Studio 2012 — Redistributable Package Requirement

You must have Visual C++ Redistributable for Visual Studio 2012 Update 3 installed on the machine where you are *running* a C++ application built the *release* libraries of the following RTI architecture packages:

- i86Win32VS2012 built with dynamic libraries
- x64Win64VS2012 built with dynamic libraries
- i86Win32dotnet4.5 and x64Win64dotnet4.5

Visual C++ Redistributable for Visual Studio 2012 Update 3 can be obtained from this Microsoft website: http://www.microsoft.com/en-ca/download/details.aspx?id=30679

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 Connext Core Libraries and Utilities User's Manual* and *Platform Notes* for more information on compilers and linkers.

- Table 1.3, "AIX Platforms," on page 1-6
- Table 1.4, "INTEGRITY Platforms," on page 1-6
- Table 1.5, "Linux Platforms on ARM CPUs," on page 1-6
- Table 1.6, "Linux Platforms on Cell BE CPUs," on page 1-6
- Table 1.7, "Linux Platforms on Intel CPUs," on page 1-6
- Table 1.8, "Linux Platforms on PowerPC CPUs," on page 1-7
- Table 1.9, "LynxOS Platforms," on page 1-8
- Table 1.10, "Mac OS Platforms," on page 1-8
- Table 1.11, "QNX Platforms," on page 1-8
- Table 1.12, "Solaris Platforms," on page 1-8
- Table 1.13, "VxWorks Target Platforms," on page 1-8
- Table 1.14, "Windows Platforms," on page 1-10

1.2 Disk and Memory Usage

Disk usage for a typical host-only installation is approximately 385 MB on Linux systems and 625 MB on Windows systems. Each additional architecture (host or target) requires an additional 162 MB on Linux systems and 402 MB on Windows systems.

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 built-in UDP/IPv4 and shared-memory[1] transport plug-ins.
- A built-in IPv6 transport is available (disabled by default) for some platforms.
- A TCP transport is also available (but is not a *built-in* transport) for some platforms.

1. The shared-memory transport is not supported on VxWorks 5.5 platforms.

See the *RTI Connext Core Libraries and Utilities Platform Notes* for details on which platforms support the IPv6 and TCP transports.

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

Table 1.3 **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
	POWER5 (64-bit mode)	IBM XLC for AIX v9.0	64p5AIX5.3xlc9.0
AIX 7.1	POWER class (64-bit mode)	IBM xlC_r for AIX v12.1	64p7AIX7.1xlc12.1
		IBM Java 1.7	64p7AIX7.1xlc12.1jdk

Table 1.4 **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]	p4080 (based on e500mc core)	Multi 6.1	GHNet2 v2	p4080Integrity10.0.2.xes-p4080-smp[3]
	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.
3. Only C and C++ APIs are supported.

Table 1.5 **Linux Platforms on ARM CPUs**

Operating System	CPU	Compiler	RTI Architecture Abbreviation
Raspbian Wheezy 7.0 (3.x kernel)	ARMv6	gcc 4.7.2[1]	armv6vfphLinux3.xgcc4.7.2
		Java Platform, Standard Edition JDK 1.7	armv6vfphLinux3.xgcc4.7.2jdk

1. Requires Linaro Gnueabihf Cross Compiler

Table 1.6 **Linux Platforms on Cell BE CPUs**

Operating System	CPU	Compiler	RTI Architecture Abbreviation
Fedora 12 (2.6.32 kernel) *Available upon request only.*	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 1.7 **Linux Platforms on Intel CPUs**

Operating System	CPU	Compiler	RTI Architecture Abbreviation
CentOS 5.4, 5.5 (2.6 kernel)	x86	gcc 4.1.2	i86Linux2.6gcc4.1.2
		Java Platform, Standard Edition JDK 1.7	i86Linux2.6gcc4.1.2jdk
	x64	gcc 4.1.2	x64Linux2.6gcc4.1.2
		Java Platform, Standard Edition JDK 1.7	x64Linux2.6gcc4.1.2jdk

Table 1.7 **Linux Platforms on Intel CPUs**

Operating System	CPU	Compiler	RTI Architecture Abbreviation
CentOS 6.0, 6.2-6.4 (2.6 kernel)	x86	gcc 4.4.5	i86Linux2.6gcc4.4.5
		Java Platform, Standard Edition JDK 1.7	i86Linux2.6gcc4.4.5jdk
	x64	gcc 4.4.5	x64Linux2.6gcc4.4.5
		Java Platform, Standard Edition JDK 1.7	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 4.0 (2.6 kernel)	x86	gcc 3.4.3	i86Linux2.6gcc3.4.3
	x64	gcc 3.4.5	x64Linux2.6gcc3.4.5
Red Hat Enterprise Linux 5.0 (2.6 kernel)	x86	gcc 4.1.1	i86Linux2.6gcc4.1.1
		Java Platform, Standard Edition JDK 1.7	i86Linux2.6gcc4.1.1jdk
	x64	gcc 4.1.1	x64Linux2.6gcc4.1.1
		Java Platform, Standard Edition JDK 1.7	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
		Java Platform, Standard Edition JDK 1.7	i86Linux2.6gcc4.1.2jdk
	x64	gcc 4.1.2	x64Linux2.6gcc4.1.2
		Java Platform, Standard Edition JDK 1.7	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
		Java Platform, Standard Edition JDK 1.7	i86Linux2.6gcc4.1.2jdk
Red Hat Enterprise Linux 6.0-6.4 (2.6 kernel)	x86	gcc 4.4.5	i86Linux2.6gcc4.4.5
		Java Platform, Standard Edition JDK 1.7	i86Linux2.6gcc4.4.5jdk
	x64	gcc 4.4.5	x64Linux2.6gcc4.4.5
		Java Platform, Standard Edition JDK 1.7	x64Linux2.6gcc4.4.5jdk
SUSE Linux Enterprise Server 11 SP2 (2.6 kernel)	x64	gcc 4.3.4	x64Linux2.6gcc4.3.4
		Java Platform, Standard Edition JDK 1.7	x64Linux2.6gcc4.3.4jdk
SUSE Linux Enterprise Server 11 SP2 (3.x kernel)	x86	gcc 4.3.4	i86Linux3gcc4.3.4
		Java Platform, Standard Edition JDK 1.7	i86Linux3gcc4.3.4jdk
Ubuntu Server 12.04 LTS	x86	gcc 4.6.3	i86Linux3.xgcc4.4.3
		Java Platform, Standard Edition JDK 1.7	i86Linux3.xgcc4.6.3jdk
	x64	gcc 4.6.3	x64Linux3.xgcc4.6.3
		Java Platform, Standard Edition JDK 1.7	x64Linux3.xgcc4.6.3jdk
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 1.8 **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

Table 1.8 **Linux Platforms on PowerPC CPUs**

Operating System	CPU	Compiler	RTI Architecture Abbreviation
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 1.9 **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 1.10 **Mac OS Platforms**

Operating System	CPU	Compiler	RTI Architecture Abbreviation
Mac OS X 10.8	x64	clang 4.1	x64Darwin12clang4.1
		Java Platform, Standard Edition JDK 1.7	x64Darwin12clang4.1jdk

Table 1.11 **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 1.12 **Solaris Platforms**

Operating System	CPU	Compiler	RTI Architecture Abbreviation
Solaris 2.9	x86	gcc 3.3.2	i86Sol2.9gcc3.3.2
		Java Platform, Standard Edition JDK 1.7	i86Sol2.9jdk
	UltraSPARC	CC 5.4 (Forte Dev 7, Sun One Studio 7)	sparcSol2.9cc5.4
		Java Platform, Standard Edition JDK 1.7	sparcSol2.9jdk
Solaris 2.10	UltraSPARC	gcc3.4.2	sparcSol2.10gcc3.4.2
		Java Platform, Standard Edition JDK 1.7	sparcSol2.10jdk
	UltraSPARC (with native 64-bit support)	gcc3.4.2	sparc64Sol2.10gcc3.4.2
		Java Platform, Standard Edition JDK 1.7	sparc64Sol2.10jdk

Table 1.13 **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

Table 1.13 **VxWorks Target Platforms**[1]

Operating System	CPU	Compiler	RTI Architecture
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
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

Table 1.13 **VxWorks Target Platforms**[1]

Operating System	CPU	Compiler	RTI Architecture
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 6.9.3.2[3]	x64	gcc 4.3.3	For Kernel Modules: pentium64Vx6.9gcc4.3.3 For Real Time Processes: pentium64Vx6.9gcc4.3.3_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.
3. Available Q1 2014

Table 1.14 **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
		Java Platform, Standard Edition JDK 1.7	i86Win32jdk
Windows 7 (64-bit Edition)	x64	Visual Studio 2010	x64Win64VS2010
		Visual Studio 2010 (C++/CLI, C# 8.0 or 9.0)	x64Win64dotnet4.0
		Java Platform, Standard Edition JDK 1.7	x64Win64jdk
Windows 8 (32-bit Edition)	x86	Visual Studio 2012	x64Win64VS2012
		Visual Studio 2012	i86Win32dotnet4.5
		Java Platform, Standard Edition JDK 1.7	x64Win64jdk
Windows 8 (64-bit Edition)	x64	Visual Studio 2012	x64Win64VS2012
		Visual Studio 2012	x64Win64dotnet4.5
		Java Platform, Standard Edition JDK 1.7	x64Win64jdk
Windows Server 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
		Java Platform, Standard Edition JDK 1.7	i86Win32jdk

Table 1.14 **Windows Platforms**

Operating System	CPU	Compiler or Software Development Kit [1] [2]	RTI Architecture
Windows Server 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
		Java Platform, Standard Edition JDK 1.7	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
		Java Platform, Standard Edition JDK 1.7	x64Win64jdk
Windows Server 2012 R2 (64-bit Edition)	x64	Visual Studio 2012	x64Win64VS2012
		Visual Studio 2012	x64Win64dotnet4.5
		Java Platform, Standard Edition JDK 1.7	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
		Java Platform, Standard Edition JDK 1.7	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
		Java Platform, Standard Edition JDK 1.7	x64Win64jdk
Windows XP Professional[3] (32-bit Edition) SP 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
		Java Platform, Standard Edition JDK 1.7	i86Win32jdk
Windows XP Professional (64-bit Edition) SP 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
		Java Platform, Standard Edition JDK 1.7	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)
- Other Compatibility Issues (Section 2.6)

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.x

2.1.2.1 Large Data with Endpoint Discovery

An endpoint (*DataWriter* or *DataReader*) created with *Connext* 5.x will not be discovered by an application that uses a previous release (4.5f or lower) 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 *RTI Connext Core Libraries and Utilities Getting Started Guide Addendum for Extensible Types*[1].
- 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.x

Connext 4.5 and 5.x 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.x support RTPS 2.1. Some earlier releases (see Table 2.1) supported RTPS 2.0 or 1.2. Because these RTPS versions are incompatible with each other, applications built with *Con-*

1. RTI_CoreLibrariesAndUtilities_GettingStarted_ExtensibleTypesAddendum.pdf

next/RTI Data Distribution Service 4.2e and higher will not interoperate with applications built with *RTI Data Distribution Service* 4.2c or lower.

Table 2.1 **RTPS Versions**

	RTPS Version
RTI Connext 4.5f and higher	2.1
RTI Data Distribution Service 4.2e - 4.5e	2.1
RTI Data Distribution Service 4.2c	2.0
RTI Data Distribution Service 4.2b and lower	1.2

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 Service 4.4d 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.x

This section points out important differences in *Connext* 5.x that may require changes on your part when upgrading from 4.5f or lower to 5.x

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

For QNX architectures, in release 5.x: The C++ libraries are now built *without* the **-fno-rtti** flag and *with* the **-fexceptions** flag. To build QNX architectures with release 5.x, 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 Custom Content Filters API

Starting with *Connext* 5.0.0, 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 in Connext 5.0.0

The *rtiddsgen*-generated type-support code for user-defined data type changed in 5.0.0 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.

2.2.2.4 Changes in Generated Type Support Code in Connext 5.1.0

The *rtiddsgen*-generated type-support code for user-defined data type changed in 5.1.0 to facilitate some new features. If you have code that was generated with *rtiddsgen* 5.0.0 or lower, you must regenerate that code using the version of *rtiddsgen* provided with this release.

2.2.2.5 New Default Value for DomainParticipant Resource Limit, participant_property_string_max_length

Starting with *Connext* 5.1.0, the default value of **participant_property_string_max_length** in the DomainParticipantResourceLimitsQosPolicy has been changed from 1024 characters to 2048 to accommodate new system properties (see Section 8.7, *System Properties*, in the *RTI Core Libraries and Utilities User's Manual*.)

2.2.2.6 New Default Value for DomainParticipant's participant_name.name

Starting with *Connext* 5.1.0, the default value for **participant_qos.participant_name.name** has been changed from the string "[ENTITY]" to NULL to provide consistency with the default name of other entities such as *DataWriters* and *DataReaders*.

2.2.2.7 Constant DDS_AUTO_NAME_ENTITY no Longer Available

Starting with *Connext* 5.1.0, the constant DDS_AUTO_NAME_ENTITY, which was used to assign the name "[ENTITY]" to a participant, has been removed from *Connext*. References to this constant must be removed from *Connext* applications.

2.2.3 Release-Specific Information for RTI Data Distribution Service 4.x, Connext 4.5 and 5.x

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 Types 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):

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

- *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.

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.

❑ **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

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

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 17.2, *How to Load XML-Specified QoS Settings*, in the *RTI Connext 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 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

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

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.x includes partial support for the "Extensible and Dynamic Topic Types for DDS" (DDS-XTypes) specification[1] 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 *RTI Connext Core Libraries and Utilities Getting Started Guide Addendum for Extensible Types*[2].

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 also need MySQL ODBC 5.1.6 (or higher). For non-Windows platforms, UnixODBC 2.2.12 (or higher) is also required.

To see if a specific architecture has been tested with the Durable Writer History and Durable Reader State features, see the *RTI Core Libraries and Utilities Platform Notes*[3].

For more information on database setup, please see the *Addendum for Database Setup*[4].

2.5 Transport Compatibility

2.5.1 Shared-Memory Transport Compatibility for Connext 4.5f and Connext 5.x

The shared-memory transport in *Connext* 4.5f and higher 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.

1. http://www.omg.org/spec/DDS-XTypes/
2. RTI_CoreLibrariesAndUtilities_GettingStarted_ExtensibleTypesAddendum.pdf
3. RTI_CoreLibrariesAndUtilities_PlatformNotes.pdf
4. RTI_CoreLibrariesAndUtilities_GettingStarted_DatabaseAddendum.pdf

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

2.5.2 Transport Compatibility for Connext 5.1.0

2.5.2.1 Changes to message_size_max

In *Connext* 5.1.0, the default **message_size_max** for the UDPv4, UDPv6, TCP, Secure WAN, and shared-memory transports changed to provide better out-of-the-box performance. Consequently, *Connext* 5.1.0 is not out-of-the-box compatible with applications running older versions of *Connext* or *RTI Data Distribution Service*.

To guarantee that communication between two applications always occurs: for a given transport, keep a consistent value for **message_size_max** in all applications within a *Connext* system.

How to Change Transport Settings in Connext 5.1.0 Applications for Compatibility with Connext 5.0.0:

If you need compatibility with a previous release, you can easily revert to the transport settings used in *Connext* 5.0.0. The new built-in **Baseline.5.0.0** QoS profile contains all of the default QoS values from *Connext* 5.0.0. Therefore, using it in a *Connext* 5.1.0 application will ensure that *Connext* 5.0.0 and 5.1.0 applications have compatible transport settings. Below is an example of how to inherit from this profile when configuring QoS settings:

```
<qos_profile name="MyProfile" base_name="BuiltinQosLib::Baseline.5.0.0">
   ...
</qos_profile>
```

How to Change message_size_max in Connext 5.0.0 Applications for Compatibility with Connext 5.1.0:

The transport configuration can be adjusted programmatically or by using XML configuration. The following XML snippet shows how to change **message_size_max** for the built-in UDPv4 transport to match the *Connext* 5.1.0 default setting of 65,507:

```
<participant_qos>
  <property>
    <value>
      <element>
        <name>dds.transport.UDPv4.builtin.parent.message_size_max</name>
        <value>65507</value>
      </element>
    </value>
  </property>
</participant_qos>
```

See Chapter 15, Transport Plugins, in the *RTI Core Libraries and Utilities User's Manual* for more details on how to change a transport's configuration.

To help detect misconfigured transport settings, *Connext* 5.1.0 will send the transport information, specifically the **message_size_max**, during participant discovery. Sharing this information will also make it easier for tools to report on incompatible applications in the system.

If two *Connext* 5.1.0 *DomainParticipants* that discover each other have a common transport with different values for **message_size_max**, *Connext* will print a warning message about that condition. Notice that older *Connext* applications do not propagate transport information, therefore this checking is not done.

You can access a remote *DomainParticipant's* transport properties by inspecting the new **transport_info** field in the DDS_ParticipantBuiltinTopicData structure. See Chapter 16, Built-in Topics, in the *RTI Core Libraries and Utilities User's Manual* for more details about this field. There

is a related new field, **transport_info_list_max_length**, in the DomainParticipantResourceLimitsQosPolicy. See Table 8.12 in the *RTI Core Libraries and Utilities User's Manual* for more details about this field.

Table 2.2 and Table 2.3 show the new default transport settings.

Table 2.2 **UDPv4, UDPv6, WAN, and TCP**

	Old Default (bytes)	New Default (bytes)	
		Non-INTEGRITY Platforms	INTEGRITY Platforms[1]
message_size_max	9,216	65,507[2]	9,216
send_socket_buffer_size	9,216	131,072	131,072
recv_socket_buffer_size	9,216	131,072	131,072

1. Due to limits imposed by the INTEGRITY platform, the new default settings for all INTEGRITY platforms are treated differently than other platforms. Please see the *RTI Core Libraries and Utilities Platform Notes* for more information on the issues with increasing the **message_size_max** default values on INTEGRITY platforms. Notice that interoperation with INTEGRITY platforms will require updating the transport property **message_size_max** so that it is consistent across all platforms.
2. The value 65507 represents the maximum user payload size that can be sent as part of a UDP packet.

Table 2.3 **Shared Memory**

	Old Default (bytes)	New Default (bytes)	
		Non-INTEGRITY Platforms	INTEGRITY Platforms[1]
message_size_max	9,216	65,536	9,216
received_message_count_max	32	64	8
receive_buffer_size	73,728	1,048,576	18,432

1. Due to limits imposed by the INTEGRITY platform, the new default settings for all INTEGRITY platforms are treated differently than other platforms. Please see the *RTI Core Libraries and Utilities Platform Notes* for more information on the issues with increasing the **message_size_max** default values on INTEGRITY platforms. Notice that interoperation with INTEGRITY platforms will require updating the transport property **message_size_max** so that it is consistent across all platforms.

2.5.2.2 Changes to Peer Descriptor Format

In *Connext* 5.1.0, the way in which the user provides a participant ID interval has changed from **[a,b]** to **[a-b]**.

2.6 Other Compatibility Issues

2.6.1 ContentFilteredTopics

Connext 5.1.0 includes a change in the generated typecode name when *rtiddsgen's* **-package** option is used in Java; this change introduces the following backward-compatibility issue.

DataWriters in 4.5x and below will not be able to perform writer-side filtering for *Connext* 5.1.0 *DataReaders* using ContentFilteredTopics. You will get a ContentFilteredTopic (CFT) compilation error message on the *DataWriter* side. Notice that this compatibility issue does not affect correctness, as the filtering will be done on the *DataReader* side.

3 What's Fixed in 5.1.0

This section includes:

3.1 Fixes to Code Generator (rtiddsgen)

3.1.1 Incorrect Typecode Name when Using rtiddsgen -package

When using *rtiddsgen* with the **-package** option in Java, the typecode name included the package name. This is inconsistent with the typecode name in C++, which does not include the package prefix. Starting with this release, the package name will no longer be included in the typecode name.

This fix introduces a backward-compatibility issue with 4.5x *DataWriters;* see ContentFiltered-Topics (Section 2.6.1).

[RTI Issue ID CODEGEN-28]

3.1.2 IDL Filenames with Periods or Hyphens Caused Compilation Errors in Generated C/C++ Code

If the name of an IDL file contained a period or hyphen (such as **msg.one.idl** or **msg-one.idl**), the generated C/C++ code failed to build. This problem has been resolved.

[RTI Issue ID CODEGEN-349]

3.1.3 .NET Code Generation for Arrays of Sequences was not Supported

.NET Code Generation for arrays of sequences of primitive or complex types was not supported. For example:

```
struct MyStruct {
    sequence<long> myLongSeqArr[3];
};
```

In some cases, *rtiddsgen* produced code that did not compile. In other cases, the deserialization of data was wrong. This problem has been resolved.

[RTI Issue ID CODEGEN-494]

3.1.4 Value of Copy Directives in Included IDL Files Incorrectly Copied to Main IDL Files

The value of copy directives in included IDL files was incorrectly copied into the main IDL files. For example, assume the following two IDL files:

Base.idl

```
//@copy-declaration #include "Other.h"
struct MyBase
{
    long m1;
};
```

Derived.idl

```
#include "Base.idl"
struct MyDerived : MyBase
{
    long m2;
};
```

Running *rtiddsgen* on **Derived.idl** file as follows:

```
rtiddsgen -language C++ Derived.idl
```

resulted in the value of the copy directive in **Base.idl** being copied into the file **Derived.h**:

```
/*
  WARNING: THIS FILE IS AUTO-GENERATED. DO NOT MODIFY.
  This file was generated from extended.idl using "rtiddsgen".
  The rtiddsgen tool is part of the RTI Connext distribution.
  For more information, type 'rtiddsgen -help' at a command shell
  or consult the RTI Core Libraries and Utilities User's Manual.
*/

#ifndef extended_53651664_h
#define extended_53651664_h

#ifndef NDDS_STANDALONE_TYPE
    #ifdef __cplusplus
        #ifndef ndds_cpp_h
            #include "ndds/ndds_cpp.h"
        #endif
    #else
        #ifndef ndds_c_h
            #include "ndds/ndds_c.h"
        #endif
    #endif
#else
    #include "ndds_standalone_type.h"
#endif

#include "Other.h"
#include "base.h"
...
```

The statement **'#include "Other.h"'** should not be in **Derived.h**. This problem has been resolved.

[RTI Issue ID CODEGEN-507]

3.1.5 Wrong Default Value for Union Members in C/C++

Unions with signed integer or boolean discriminators that did not have a default case value were initialized with incorrect values.

For example, in the following union the default discriminator value should be -200 (the lowest value associated with any member). However the value that was used to initialize the union was 300.

```
union MyUnion switch (long) {
    case -200:
        long m1;
    case 300:
        long m2;
};
```

As another example, the default discriminator value for the following union should be TRUE, but it was initialized to FALSE.

```
union MyUnion switch (boolean) {
    case TRUE:
        long m1;
};
```

This problem has been resolved.

[RTI Issue ID CODEGEN-519]

3.1.6 Comparing Typecode Generated by C/C++ and Java Applications Returned False if Typecode Contained Unions with Default Discriminator s

Comparing a typecode generated by a C/C++ application with the same typecode generated by a Java application returned false if the typecode contained unions with default discriminators, such as:

```
union MyUnion switch(long) {
    case 0:
        long m1;
    default:
        long m2;
};
```

If a Java application reading the Publication or Subscription built-in topics discovered a C/C++ *DataWriter/DataReader* using the above type and compared the remote *DataReader/DataWriter's* typecode with the locally created typecode (using the **DDS_TypeCode_equal()** operation), the result was false when it should have been true. This problem has been resolved.

[RTI Issue ID CODEGEN-520]

3.1.7 Typecodes Generated for Enumeration Types did not Include Extensibility Information

The **DDS_TypeCode_extensibility_kind()** operation always returned DDS_EXTENSIBLE_-EXTENSIBILITY, even if the enumeration was marked as FINAL or MUTABLE. For example:

```
enum MyEnum {
    ENUM_1,
    ENUM_2
}; //@Extensibility FINAL_EXTENSIBILITY
```

If **DDS_TypeCode_extensibility_kind()** was called on the typecode generated by *rtiddsgen* for the above enumeration, the result was DDS_EXTENSIBLE_EXTENSIBILITY, when it should have been DDS_FINAL_EXTENSIBILITY. This problem has been resolved.

[RTI Issue ID CODEGEN-525]

3.1.8 Incorrect Extensibility Information for Generated Typecodes if Extensibility Annotation not Explicitly Used

The code generator may have generated typecodes with an incorrect extensibility kind if the extensibility annotation was not used explicitly in the types declared in an IDL file. For example:

```
enum MyEnum {
    ENUM_1,
    ENUM_2
};

struct MyStruct {
    long m1;
    MyEnum m2;
}; //@Extensibility MUTABLE_EXTENSIBILITY
```

In the above example, the typecode for MyEnum had MUTABLE extensibility when it should have had EXTENSIBLE extensibility. (Due to a bug in the code generator, the enumeration mistakenly inherited the extensibility from the structure that followed it.) This problem has been resolved.

[RTI Issue ID CODEGEN-526]

3.1.9 Deserialization Error when Subscribing to Extended Valuetype and Receiving Samples from Base Valuetype

A *DataReader* subscribing to an extended valuetype failed to deserialize samples from *DataWriter* publishing a base valuetype. For example:

```
valuetype BaseValue {
    public long m1;
}; //@Extensibility EXTENSIBLE_EXTENSIBILITY

valuetype ExtendedValue : BaseValue  {
    public string<128> m2;
}; //@Extensibility EXTENSIBLE_EXTENSIBILITY
```

A *DataReader* subscribing to ExtendedValue failed to deserialize samples from a *DataWriter* publishing BaseValue.

[RTI Issue IDs CODEGEN-527 and CODEGEN-528]

3.1.10 Deserialization Error when Subscribing to Extended Type where Last Member Requires 8-byte Alignment

A *DataReader* or Dynamic *DataReader* subscribing to an extended type may have failed to deserialize samples from a *DataWriter* publishing a base type if the last member of the extended type was a long long, unsigned long long, double, or long double. For example:

```
struct MyBaseType {
    char m1;
};
struct MyExtendedType {
    char m1;
    long m2;
};
```

In the above example, a *DataReader* subscribing to MyExtendedType failed to deserialize samples coming from a *DataWriter* publishing MyBaseType. This issue has been resolved.

[RTI Issue ID CODEGEN-530 and CORE-5536]

3.1.11 XML Files Generated with -convertXml Option Incompatible with Previous Releases if They Contained Enumerations

The XML files generated using *rtiddsgen's* **-convertXml** option were incompatible with previous releases (4.5f and below) if they contained enumerations. For example:

```
enum MyEnum {
    ENUM_1,
    ENUM_2
};
```

For the above enumeration, *rtiddsgen* generated the following XML:

```
<enum name="MyEnum" bitBound="32">
    <enumerator name="ENUM_1"/>
    <enumerator name="ENUM_2"/>
</enum>
```

The XML parser of older releases (4.5f and below) did not understand the attribute **bitBound** and therefore failed to parse the generated XML file. This problem has been resolved.

[RTI Issue ID CODEGEN-532]

3.1.12 Incorrect Keyhash Generated when Type of Key Members was Structure Inheriting from Another Structure—C and C++ Only

Connext generated an incorrect keyhash value for samples whose type contained a key member where the type was a structure inheriting from another structure. For example:

```
struct BaseStruct {
    long nonKey;
    long key; //@key
};

struct DerivedStruct: BaseStruct {
    long nonKeyDerived;
    long keyDerived; //@key
};

struct NestedKeyStruct {
    long nonKey;
    DerivedStructKeyhash key; //@key
};
```

In the above example, the keyhash from samples with type NestedKeyStruct was generated incorrectly. Consequently, you may have seen samples from different instances as part of the same instance. This problem only affected C and C++ code generation.

[RTI Issue ID CODEGEN-535]

3.1.13 Invalid Value for max_blocking_time Tag in Generated USER_QOS_PROFILES.xml

When *rtiddsgen* is used with the **-example** command-line option, it generates an example QoS profile file called USER_QOS_PROFILES.xml. In this file, the <max_blocking_time> was set to this value:

```
<max_blocking_time>
    <sec>60</sec>
</max_blocking_time>
```

Although the intent was to set max_blocking_time to 60 seconds, the actual value was INFINITE because the XML file did not set the tag <nanosec> under <max_blocking_time> and the default value for <nanosec> is INFINITE.

This problem has been resolved by explicitly setting <nanosec> to 0, as follows:

```
<max_blocking_time>
    <sec>60</sec>
    <nanosec>0</nanosec>
</max_blocking_time>
```

[RTI Issue ID CODEGEN-566]

3.1.14 Extensibility Annotation not Supported in XSD Type Representation if Type was Valuetype

Extensibility annotation was not supported in an XSD type representation if the type was a valuetype.

For example, providing the following XSD type as an input to the code generator generated errors:

```
<?xml version="1.0" encoding="UTF-8"?>
<types xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:noNamespaceSchemaLocation="/Users/fernando/RTI/Projects/GE/tmp
/codegen583/../../../XTypes/bigpine_IN/modules/nddsgen.1.0/scripts/..
/resource/rtiddsgen/schema/rti_dds_topic_types.xsd">
    <valuetype name="myVT" typeModifier="none" extensibility="mutable">
        <member name="a1" type="long" visibility="public"/>
    </valuetype>
</types>
```

Errors:

```
/var/folders/1H/1HUnA0jBF-GJnhekcv4HXU+++TM/-Tmp-/rtiddsMy-
Type.xsd4410202923128610064modulesxsd.xml:3 Attribute "extensibility" must
be declared for element type "valuetype".
/var/folders/1H/1HUnA0jBF-GJnhekcv4HXU+++TM/-Tmp-/rtiddsMy-
Type.xsd4410202923128610064modulesxsd.xml:3 Attribute "extensibility" must
be declared for element type "valuetype".
```

This problem has been resolved.

[RTI Issue ID CODEGEN-580]

3.1.15 Error Converting to XML using -convertToXml if IDL Contained Valuetype with Extensibility Annotation

You may have seen an error when converting IDL to XML with *rtiddsgen* if the IDL included a valuetype with the extensibility annotation.

For example, given the following IDL:

```
valuetype MyVT {
    public long a1;
}; //@Extensibility EXTENSIBLE_EXTENSIBILITY
```

If you ran *rtiddsgen* with the **-convertToXML** option, *rtiddsgen* generated the following error message:

```
file:///local/preship/ndds/ndds.5.0.0/scripts/../resource/rtiddsgen/xml/
simplifiedXmlTransform.xsl; Line #46; Column #59; Cannot add attribute
extensibility after child nodes or before an element is produced. Attri-
bute will be ignored.
```

The XML output file was generated, but it did not contain the extensibility information:

```
<?xml version="1.0" encoding="UTF-8"?>
```

```
<types xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
 xsi:noNamespaceSchemaLocation="/local/preship/ndds/ndds.5.0.0/scripts/../
 resource/rtiddsgen/schema/rti_dds_topic_types.xsd">
    <valuetype name="MyVT" typeModifier="none">
        <member name="a1" type="long" visibility="public"/>
    </valuetype>
</types>
```

A workaround to this problem was to use structures instead of valuetypes, since they are equivalent. Also, if the extensibility value was EXTENSIBLE_EXTENSIBILIY, you could have removed the directive from the IDL file because EXTENSIBLE_EXTENSIBILIY is the default value. These workarounds are no longer necessary, as this problem has been resolved.

[RTI Issue ID CODEGEN-583]

3.1.16 Possible Incorrect Default Value for Type Members not Received on Wire—Java and .NET Only

With the introduction of the Extensible types specification, it is possible to publish a base type and subscribe to a derived or extended type. For example:

Base type:

```
struct MyBaseType {
    long m1;
};
struct MyDerivedType {
    long m1;
    long m2;
};
```

If a *DataReader* subscribing to the above derived type receives a sample from a *DataWriter* publishing the base type, the fields not present on the base type must be initialized to a well-defined default value. For primitive types, this default value is 0.

However, due to a bug in the initialization of the members not present on the wire, the field **m2** above may have gotten a value other than 0 in the previous release. This problem, which only affected Java and .NET code generation, has been resolved.

[RTI Issue ID CODEGEN-584]

3.1.17 Generated Equal Method in Java Type Returned Wrong Results in Some Cases

The **equal** method in a Java type may have returned true even if the two types were not equal, if the types contained members that were arrays of sequences.

For example, with the following type, calling the **equal** method on two samples, where the first sample did not set any elements in the sequence and the second one did, returned true:

```
struct MyType {
    sequence<long> m1[2];
};
```

This problem has been resolved.

[RTI Issue ID CODEGEN-585]

3.1.18 Java Deserialization of Sequences with Length Exceeding Maximum did not Produce Error

Deserialization of sequences whose length exceeded the maximum allowed did not produce an error on the *DataReader* side.

For example, consider the following two IDL types:

```
struct MyBigSeqType {
    sequence<long,50> m1;
};

struct MySmallSeqType {
    sequence<long,20> m1;
};
```

If a *DataReader* with type MySmallSeqType received a sample from a *DataWriter* with type MyBigSeqType containing more than 20 elements, *Connext* should have reported a deserialization error, but it did not. This problem has been resolved; now the error will be reported.

Note: *DataWriter*/*DataReader* matching in the above scenario is not allowed in the Extensible Types specification. However, you may have run into this bug if you set the property **dds.type_consistency.ignore_sequence_bounds** to true or if you disabled type checking.

[RTI Issue ID CODEGEN-590]

3.1.19 Error from rtiddsgen when NDDSHOME Ended with "\"—Windows Systems Only

On Windows systems, if the NDDSHOME environment variable was set to a path name that ended with a backwards slash "\", *rtiddsgen* reported a java.lang.NoClassDefFoundError error. This problem has been resolved.

[RTI Issue ID CODEGEN-600]

3.1.20 Pointers not Supported when Generated Code Compiled in Standalone Mode in C/C++

Pointers were not supported when generated C/C++ code was compiled in standalone mode.

For example, consider the following IDL file:

```
struct MyType {
    long * m1;
};
```

Trying to compile the generated code in standalone mode for the above type caused compilation errors in the previous release. This problem has been resolved.

[RTI Issue ID CODEGEN-606]

3.1.21 Problems in copy_from and Equals methods generated for a Structure Inheriting from Another Structure—.NET API Only

The **copy_from()** and **Equals()** methods generated for a structure inheriting from another structure in the .NET API were wrong because they did not consider the base structure as part of the operation.

For example:

```
struct BaseStruct {
    long m1;
};
struct DerivedStruct : BaseStruct {
    long m2;
};
```

In the above example, the **DerivedStruct::copy_from()** method did not copy the base member **m1**. The **DerivedStruct::Equals()** method did not compare the value of the base member **m1**.

The **copy_from method()** method is used by the *DataReader*'s take and read operations when they copy the received data. Therefore, the received data may have been incorrect since the base portion of the type was not copied. This problem has been resolved.

[RTI Issue ID CODEGEN-565, CODEGEN-615]

3.1.22 Wrong Makefile Generated for Java if IDL Included @copy-java Before Any Type Declarations

Specifying the "@copy-java" directive before any type declarations in an IDL file resulted in incorrect makefile generation for the Java language. This issue has been resolved.

[RTI Issue ID CODEGEN-591]

3.1.23 Generated C++ Code did not Add 'LL' Suffix to long long Literals

C++ code generated using *rtiddsgen* did not add an "LL" suffix to long long literals. This problem has been resolved.

[RTI Issue ID CODEGEN-598]

3.1.24 DataReader could Provide Samples with Invalid Values for Enumeration Fields

A *DataReader* subscribing to a topic for which the type is extensible or mutable, and where the last member is an enumeration, may have provided samples to the application in which the enumeration value was invalid. This may have occurred if a *DataWriter* published a compatible type in which the same enumeration had additional values. For example:

DataWriter type:

```
enum MyEnum {
 ENUM_1,
 ENUM_2,
 ENUM_3
};

struct MyStruct {
  MyEnum m1;
};
```

DataReader type:

```
enum MyEnum {
 ENUM_1,
 ENUM_2
};

struct MyStruct {
  MyEnum m1;
};
```

In the above example, it was possible for the *DataWriter* to send a sample where the value of **m1** was ENUM_3. When the *DataReader* received that sample, it should report a deserialization error and discard the sample because it does not recognize ENUM_3. However, due to this bug the *DataReader* assigned ENUM_3 to the enumeration value. This issue has been resolved.

[RTI Issue ID CODEGEN-622]

3.1.25 Serialization of Optional Members in Extensible Types Possibly Wrong in C/C++ and Java

Serialization of optional members in extensible types may have been wrong in C/C++ and Java if the member was a non-primitive member whose serialized size was greater than 65535 bytes. Consequently, this would cause *DataReaders* to fail to deserialize incoming samples.

For example:

```
struct MyStruct {
    char payload[80000]; //@Optional
}; //@Extensibility EXTENSIBLE_EXTENSIBILITY
```

Samples from the above type would not have been serialized correctly. Notice that the problem did not affect data structures marked as MUTABLE. For example:

```
struct MyStruct {
    char payload[80000]; //@Optional
}; //@Extensibility MUTABLE_EXTENSIBILITY
```

This issue has been resolved.

[RTI Issue ID CODEGEN-624]

3.1.26 .NET Deserialization Error when Subscribing to Extended Type and Receiving Samples from Base Type

A .NET *DataReader* subscribing to an extended type failed to deserialize samples from a *DataWriter* publishing a base type when the first field in the extended type that is not present in the base type was non-primitive. For example:

```
struct BaseValue {
    long m1;
}; //@Extensibility EXTENSIBLE_EXTENSIBILITY
struct ExtendedValue : BaseValue  {
    long m2[2];
}; //@Extensibility EXTENSIBLE_EXTENSIBILITY
```

A *DataReader* subscribing to an ExtendedValue failed to deserialize samples from *DataWriter* publishing BaseValue.

[RTI Issue ID CODEGEN-626]

3.1.27 Multidimensional Arrays of Enumerations not Supported in .NET API

Declaring multidimensional arrays of enumerations in IDL resulted in the following error when trying to send data using a .NET *Connext* application:

```
enum MyEnum {
    ENUM_1,
    ENUM_2
};

struct MyStruct {
    MyEnum m1[2][2];
};

at DDS.CdrStream.serialize_enum_array(Array elems, Int32 total_length) in
c:\ndds_head\modules\dds_dotnet.1.0\srccpp\managed\managed_cdr.cpp:line
1183 at XTypeBasePlugin.serialize(TypePluginDefaultEndpointData
endpoint_data, XTy peBase sample, CdrStream& stream, Boolean
serialize_encapsulation, UInt16 encapsulation_id, Boolean serialize_sample,
Object endpoint_plugin_qos) in c:\ndds_head\modules\nddsgen.1.0\src-
cpp\xtype\xtypeplugin.cpp:line 18633 at DDS.TypePlu-
gin`4.serialize_forwarder(Void* endpoint_data, Void* sample, RTICdrStream*
stream, Int32 serialize_encapsulation, UInt32 encapsulation_id, Int32
serialize_sample, Void* endpoint_plugin_qos) in c:\ndds_head\mod-
ules\dds_dotnet.1.0\srccpp\managed\managed_data.cpp:line 685
PRESWriterHistoryDriver_initializeSample:!serialize
WriterHistoryMemoryPlugin_addEntryToSessions:!initialize sample
WriterHistoryMemoryPlugin_getEntry:!add virtual sample to sessions
```

```
WriterHistoryMemoryPlugin_addSample:!get entry
PRESWriterHistoryDriver_addWrite:!add_sample
PRESPsWriter_writeInternal:!collator addWrite
```

This problem has been resolved.

[RTI Issue ID CODEGEN-632]

3.1.28 NoClassDefFoundError when using ndds_standalone_type.jar

You may have seen the following run-time error when using the **ndds_standalone_type.jar** file and working with sequences:

```
java.lang.NoClassDefFoundError: com/rti/dds/util/Sequences
```

This problem has been resolved.

[RTI Issue ID CODEGEN-641]

3.1.29 Incorrect Enumerator Values when Parsing XML Enumeration

In previous releases the value assigned to enumerators when declaring an enumeration in XML may have been incorrect. This occurred when the user provided explicit values for some of the enumerators. For example:

```
<enum name="MyEnum">
    <enumerator name="Enumerator0" value="100"/>
    <enumerator name="Enumerator1"/>
    <enumerator name="Enumerator2"/>
</enum>
```

In the above example, Enumerator1 should have the value 101. However, in previous releases the value was 0. This problem has been resolved.

[RTI Issue ID CORE-6027]

3.2 Fixes Related to Content Filtering

3.2.1 Incorrect Results from Evaluation of Filter Expressions with Long Long Members

Evaluation of filter expressions that referred to 64-bit integers (long long or unsigned long long) members may have had erroneous results. Only comparisons between 64-bit integers of the same type were guaranteed to produce the correct result. For example, comparing any 64-bit integer with a 32-bit integer may have produced incorrect results.

This problem is resolved. However, you must take care to use the correct signedness. For example, the SQL filter compiler will accept a statement comparing an 'unsigned long long' to a negative signed 'long', but this may yield an incorrect result during the compare.

See also: Subscribing Applications Required More CPU (Section 3.11.3)

[RTI Issue ID CORE-3416]

3.2.2 Incorrect Results from Evaluation of Filter Expressions with Float Members

Evaluation of filter expressions that referred to float members may have had erroneous results. For example:

```
struct MyType {
    float m1;
};
```

Evaluating the SQL filter expression "m1 = 4.1" for the above type may have caused an erroneous result. Therefore a sample that should have passed a filter may have been filtered out.

This issue has been corrected by demoting the operands to float if at least one operand is float.

See also: Possible Segmentation Fault when property_list_max_length Fields Set to 0 (Section 3.14.2)

[RTI Issue ID CORE-5529]

3.2.3 DataReaders using ContentFilteredTopic Received Samples that should have been Filtered Out

DataReaders using a content filter may have received samples that should have been filtered out. This problem only existed when all of the following conditions were met:

- ❏ Communication was reliable.
- ❏ The *DataWriters* sending samples to the *DataReader* were configured for writer-side filtering.
- ❏ The *DataWriters* discovered the *DataReader* while the application was blocked on the write operation waiting for samples to be acknowledged in the writer's queue.

This problem is resolved.

[RTI Issue ID CORE-5530]

3.2.4 QueryCondition's get_query_parameters() was not Thread Safe

Calling QueryCondition's **get_query_parameters()** from one thread while setting the parameters from a different thread was not safe and may have caused a segmentation fault. This problem has been resolved.

[RTI Issue ID CORE-5554]

3.2.5 Samples Incorrectly Filtered Out during Retransmission for Some DataReaders using ContentFilteredTopics

In some scenarios, samples could have been incorrectly filtered out when trying to repair a sample for a *DataReader* with a ContentFilteredTopic or when trying to send historical data to a late-joining *DataReader* with a ContentFilteredTopic.

Please see Chapter 10, Reliable Communications, in the *RTI Connext Core Libraries and Utilities User's Manual* for details on when are samples retransmitted.

This problem has been resolved.

[RTI Issue ID CORE-5667]

3.2.6 Incorrect Parameter Sequence Passed to DDSWriterContentFilter's writer_compile()

The parameter sequence passed to the DDSWriterContentFilter's **writer_compile()** operation had the value of all the elements set equal to the value of the first element. When using a custom filter that implements DDSWriterContentFilter, this can result in incorrect filtering results. For built-in filters, this could have caused a filter expression compilation failure in DDSWriterContentFilter's **writer_compile()** operation, but it would not have prevented filtering from being performed on the *DataWriter*. This problem has been resolved.

[RTI Issue ID CORE-5700]

3.2.7 Segmentation Fault if Filter-Expression Compilation Failed

DDS_SqlFilter_writerCompile was not handling an error condition correctly, which caused a segmentation fault if compilation of a content-filter expression failed on a DataWriter. This problem has been resolved.

[RTI Issue ID CORE-5701]

3.2.8 SQL Filter did not Properly Filter Members in Structure or Valuetype using Inheritance in Some Cases

Expression evaluation using the SQL filter may have worked incorrectly if the type associated with the filter was a valuetype/structure using inheritance or contained a member that was a valuetype/structure using inheritance.

For example, consider the following IDL:

```
module Common {
    struct BaseType {
        long big_number;
        short small_number;
    };
    valuetype DerivedType : BaseType {
        public short next_number;
    };
}
```

In the above IDL, the evaluation of a content filter expression for type **DerivedType** may have caused bad behavior ranging from improper filtering to segmentation faults. This problem has been resolved.

[RTI Issue ID CORE-5770]

3.2.9 SQL Filter Expressions on Pointer Type Members may have Behaved Incorrectly

In some situations, a content filter expression referring to pointer members[1] may have incorrectly filtered out samples that should have passed the filter, or resulted in other undefined behavior.

These situations included, but were not limited to, the following:

- ❏ Pointers in a base type
- ❏ Pointers to a type that has a base

This problem has been resolved.

[RTI Issue IDs CORE-5798, CORE-5804, CORE-5816]

3.2.10 Possible Memory Corruption Provoked by SQL Fillter Compile Operation if Large Expression was Set

When a large expression was used with a SQL filter, the **compile()** operation may have corrupted the memory by using an out-of-bounds address. This issue has been resolved.

[RTI Issue ID CORE-5815]

3.2.11 Issues with SQL Filter Expressions on DynamicDataReaders with Inherited Types

Given a type T with a base class:

```
struct Base {
   // ...
};

struct T : Base {
   long x;
};
```

1. Pointers are described in the *RTI Core Libraries and Utilities Users's Manual*, Section 5.4.6.8.

In some cases, a DynamicDataReader using a ContentFilteredTopic on type T (for example, with an expression "x = 1") may have incorrectly filtered out samples that should have passed the filter. This problem has been resolved.

[RTI Issue ID CORE-5832]

3.2.12 SQL Filter did not Properly Filter Different-but-Compatible Types in Some Cases

When a *DataWriter* and *DataReader* use different-but-compatible types for the same *Topic*, the *DataReader* sets the values of any members that are missing in the samples written by the *DataWriter* to default values.[1]

The default value for an enumeration is its first constant. For example, the default value of MyEnum is ONE.

```
enum MyEnum {
    ONE = 1 // Note that 0 (zero) is not a valid value
    TWO = 2
}
```

In the previous release, SQL filters did not correctly initialize this value and would see zero. Therefore a *DataReader* with an expression such as "my_enum = 'ONE'" would miss samples from a *DataWriter* whose type did not have the member **my_enum**.

```
struct WriterType  {
    long x;
};

struct ReaderType {
    long x;
    MyEnum my_enum;
};
```

This problem has been resolved.

[RTI Issue ID CORE-5834]

3.2.13 Writer-Side Filtering Functions Invoked After Filter Unregistered

If you install a ContentFilter that implements the writer-side filtering APIs, *Connext* was able to call those APIs even after the ContentFilter had been unregistered. This problem has been resolved.

[RTI Issue ID CORE-5347]

3.3 Fixes Related to Dynamic Data

3.3.1 Dynamic Data Property Structures not Correctly Initialized in C++

The Dynamic Data structures DDS_DynamicDataProperty_t, DDS_DynamicDataTypeSerializationProperty_t, and DDS_DynamicDataTypeProperty_t were not properly initialized in C++. This problem has been resolved; all three structures now have default constructors.

[RTI Issue ID CORE-3558]

1. This is described in the *RTI Core Libraries and Utilities Getting Started Guide Addendum for Extensible Types*, Section 2. Type Safety and System Evolution)

3.3.2 Possible Segmentation Fault when Publishing Data with Dynamic DataWriter—Java API Only

Due to a problem in the *Connext* JNI layer, the garbage collector could, in rare cases, reclaim a DynamicData object that was still in use within the Dynamic Data *DataWriter's* write operation, resulting in a segmentation fault. This problem has been resolved.

[RTI Issue ID CORE-5488]

3.3.3 Dynamic DataReader Failed to Deserialize Samples in Some Cases

A Dynamic Data *DataReader* failed to deserialize samples if the underlying type of the *DataReader's Topic* contained sequences or arrays of mutable types. For example:

```
struct MyMutableType {
    long m1;
    long m2;
}; //@Extensibility MUTABLE_EXTENSIBILITY

struct MyType {
    long m1;
    sequence<MyMutableType,3> m2;
}; //@Extensibility FINAL_EXTENSIBILITY
```

This may have resulted in either a deserialization error, or no reported error but an incorrect value in the received samples.

[RTI Issue ID CORE-5497]

3.3.4 Dynamic Data Mishandled Discriminator Values

The Dynamic Data implementation mishandled union discriminator values, causing their signs to be ignored in some cases and resulting in the incorrect member of the union being set. This problem has been resolved.

[RTI Issue ID CORE-5874]

3.3.5 Dynamic Data DataReader Issued Segmentation Fault if TypeSupport Serialization Property trim_to_size was True

The serialization member of the DynamicDataProperty_t that is used to create the DynamicDataTypeSupport has a field called **trim_to_size** that is used to control the growth of the serialization object in a Dynamic Data object (for additional information see Chapter 20, *Sample Data Memory Management*, in the *RTI Connext Core Libraries and Utilities User's Manual*).

If **trim_to_size** was set to RTI_TRUE (which is not the default), in rare cases the *DataReader* may have issued a segmentation fault upon sample reception. This problem has been resolved.

[RTI Issue ID CORE-5454]

3.3.6 Errors Deserializing Extensible Types in Dynamic Data DataReaders

The samples received by a Dynamic Data *DataReader* may have been deserialized incorrectly if the *DataWriter* publishing the samples used an extensible type with a subset of the members contained in the *DataReader's* type.

For example:

```
struct DataWriterType {
    char m1;
}; //@Extensibility EXTENSIBLE_EXTENSIBILITY

struct DataReaderType {
    char m1;
    long m2;
}; //@Extensibility EXTENSIBLE_EXTENSIBILITY
```

In the above example, when the *DataReader* receives a sample, it should initialize the member **m2** to its default value since the *DataWriter* did not provide a value for it. However, if the application tried to access the **m2** value using the corresponding Dynamic Data API, the received value may have been invalid and different than the default.

This problem only occurred if the types were marked as EXTENSIBLE (as in the above example) and the *DataReader* must have been using an extended version of the *DataWriter* type. This problem has been resolved.

[RTI Issue ID CORE-5473]

3.3.7 ValueTypes or Structures Inheriting from Alias not supported in Dynamic Data

The previous release did not support using Dynamic Data to publish or subscribe to a topic whose underlying type was a structure or valuetype that inherited from an alias (typedef). For example:

```
struct MyBaseType {
    long m1;
};
typedef MyBaseType MyBaseTypeAlias;
struct MyDerivedType : MyBaseTypeAlias  {
    string<128> m2;
};
```

This problem has been resolved.

[RTI Issue ID CORE-5478]

3.3.8 Dynamic Data DataReader Reported Deserialization Errors when Receiving Strings with Length Equal to Maximum Length

A Dynamic Data *DataReader* reported deserialization errors when receiving strings with a length equal to the maximum length in the IDL file. For example:

```
struct MyType {
    string<10> myStr;
}
```

If a *DataWriter* sent a sample containing a string with length 10, the DynamicData *DataReader* failed to deserialize the sample. This problem has been resolved.

[RTI Issue ID CORE-5571]

3.3.9 Dynamic Data.get_short_array() Returned Wrong Value

The Dynamic Data.get_short_array() operation returned the wrong size for an array obtained from the Dynamic Data object. This problem has been resolved.

[RTI Issue ID CORE-5618]

3.3.10 DDS_DynamicData's get_wstring() and get_string() Failed to Report Minimum Required Size if Input Size Too Small

When DDS_DynamicData's **get_wstring()** or **get_string()** functions are called with an insufficient (but greater than zero) size, the function call should fail and the size should contain the minimum required size. In the previous release, the minimum required size was not reported. This problem has been resolved.

[RTI Issue ID CORE-5750]

3.4 Fixes Related to Asynchronous Publication and Large Data

3.4.1 DataReader Rejected Large Data Samples from Non-RTI DataWriter

A *DataReader* in a *Connext* application rejected some large data samples from a *DataWriter* in a non-RTI DDS application. This occurred due to a combination of (a) the *DataWriter* not sending a timestamp with each RTPS message containing fragments of the sample, and (b) the *DataReader* expecting a timestamp with each message because its Destination Order QoS policy was set to "by source timestamp."

If the first message with fragments of the sample received by the *DataReader* did not have a timestamp (which could happen if the multiple messages of the large sample were received out of order), the *DataReader* would not have a valid source timestamp for the sample, consequently leading the *DataReader* to reject the sample.

[RTI Issue ID CORE-5367]

3.4.2 Potential Segmentation Fault when Setting dynamically_allocate_fragmented_samples to TRUE

In certain scenarios, setting **dynamically_allocate_fragmented_samples** (in the *DataReader's* ReaderResourceLimitsQosPolicy) to TRUE caused a segmentation fault in the *Connext* application. This problem has been resolved.

[RTI Issue ID CORE-5144]

3.5 Fixes Related to Ping and Spy Utilities

3.5.1 Ping and Spy Utilities Fail if Configured via XML File with Monitoring Properties

RTI's Ping and Spy utilities (*rtiddsping* and *rtiddsspy*) do not support *RTI Monitoring Library.* If you configure these utilities with an XML file that also includes monitoring properties (rti.monitor.*), the applications may issue a segmentation fault.

This problem has been resolved. Now Ping and Spy will ignore any monitoring properties in the XML file and log this info-level message:

```
The usage of the monitoring library is not supported in this tool.
```

[RTI Issue ID CORE-5228]

3.5.2 Possible Deadlock in rtiddsspy in Very Rare Cases

It was possible for *rtiddsspy* to deadlock in very rare cases if there was no data available. This problem has been resolved.

[RTI Issue ID CORE-5363]

3.5.3 Ping and Spy Utilities Returned Error if Parameter Contained Spaces

If a parameter to *Prototyper* contained a space (even if it was surrounded by quotes), *Prototyper* reported an error message. For example, this command:

```
$ rtiddsspy -qosFile "File Name.xml"
```

returned an error such as:

```
RTI Data Distribution Service Spy: Unrecognized parameter 'Name.xml'
```

This problem has been resolved.

[RTI Issue ID DIABLO-753]

3.6 Fixes Related to Transports

3.6.1 No Data Exchange when Using Force Asynchronous and Asymmetric Mode

When using asymmetric mode and the **force_asynchronous_send** option set to true, the involved participants did not complete discovery nor exchange any user data. This problem has been resolved.

[RTI Issue ID COREPLG-176]

3.6.2 Interface Aliases not Supported in Built-in UDPv4 Transport

On UNIX-based systems, the UDPv4 transport ignored the alias IP addresses associated with a network device. A *DomainParticipant* did not announce these IP addresses to other *DomainParticipants*.

On Windows systems, *DomainParticipant* creation failed when a network device was associated with more than one IP address and was configured to use multicast for discovery.

This problem has been resolved.

[RTI Issue ID CORE-5301]

3.6.3 Different Behavior when 127.0.0.1 vs. Host IP Address Added to Initial Peers when Shared Memory Enabled

If the built-in shared memory transport was enabled but **shmem://** was left out of the initial peers list, there was different behavior depending on whether localhost or the machine's IP address was added to the initial peers list (based on the default behavior of the **ignore_loopback_interface** property for the *DomainParticipant*). Applications that put the IP address in the initial peers never completed discovery and would not communicate with each other.

This problem has been resolved. The default behavior of **ignore_loopback_interface** is now to not only check if shared memory is enabled, but it must also be in the initial peers list in order to disable UDPv4 local traffic. Otherwise, UDPv4 traffic will not be disabled simply because shared memory is enabled.

[RTI Issue ID CORE-5386]

3.6.4 UDPv4 Transport Now Supported for QNX Platforms with 50+ NICs

On QNX targets with more than ~50 network interfaces, the UDPv4 transport did not work and you may have seen this error message when the *DomainParticipant* was created:

```
RTIOsapi_getFirstValidInterface:OS ioctl(SIOCGIFCONF)() failure, error 0X4F
NDDS_Transport_UDPv4_query_interfaces:ioctl(SIOCGIFCONF) error 0X4F
RTINetioConfiguratorUtil_setupUDPv4Plugin:!create plugin
DDS_DomainParticipantConfigurator_setup_builtin_transports:!install
transport plugin aliases = builtin.udpv4
```

This release corrects this limitation.

[RTI Issue ID CORE-5387]

Latency Performance Test for C++: Possible Failure if Max Number of Subscribers Set to LATENCY_MAXIMUM_SUBSCRIPTIONS

3.6.5 Error Parsing Transport Properties when Multiple Custom Transports are Installed

When installing more than one custom transport, it was possible that some of the properties were not parsed correctly for all transports except for the first one. Depending on whether or not the property was optional, this would have resulted in an error or a partially configured transport. This problem has been resolved.

Note: The order of the transports is based on the order in which they are listed in **dds.transport.load_plugins**.

[RTI Issue ID CORE-5435]

3.6.6 Connection Lost after "wrong msg signature" Error when Using TCP Transport

When sending a message over the TCP transport, *Connext* inserts a reserved signature value in its message header. On reception, if this signature value is not found in the expected location, *Connext* generates a "wrong msg signature" error message.

In previous releases, if the transport was configured in asymmetric mode, the generation of the "wrong msg signature" error resulted in an unrecoverable connection loss.

This issue has been resolved by allowing the TCP transport to recover the connection after the error message is produced.

[RTI Issue ID COREPLG-100]

3.7 Fixes Related to Type Representation and Type Matching

3.7.1 If Two Types for Same Topic Detected as Incompatible, Warning did not Include Full Type Names

Consider the following types, used for the same Topic in a *DataWriter* and *DataReader*:

```
module module1 {
    struct WriterType {
        long a;
    };
};

module module2 {
    struct ReaderType {
        short a;
    };
};
```

Connext will detect that these types are incompatible, because their first members have incompatible types (long and short).

In the previous release, if Warning verbosity was enabled, *Connext* printed this message:

```
RTICdrTypeObjectStructureType_is_assignable:types not assignable:
structures are not assignable: WriterType, ReaderType
```

The warning did not include the module name. This problem has been resolved; the new warning will be:

```
RTICdrTypeObjectStructureType_is_assignable:types not assignable:
structures are not assignable: module1::WriterType, module2::ReaderType
```

[RTI Issue ID CORE-5603]

3.7.2 DDS_TypeCodeFactory's create_struct_tc() did not Allow Specifying Non-Default Member IDs

The members of a struct type have IDs that identify them (for example when *Connext* checks if two types are compatible). By default, *Connext* assigns these IDs automatically, but you can specify other values.

The DDS_TypeCodeFactory's **create_struct_tc()** operation creates a DDS_TypeCode representing an IDL struct type, but it was not possible to specify concrete member IDs. This problem has been resolved. Now the type DDS_StructMember has an **id** field that applications can set to specify a non-default member ID.

[RTI Issue ID CORE-5713]

3.7.3 Types with Identical Structure but Different Extensibility Kinds Incorrectly Considered Equal

Connext may have incorrectly allowed communication between two applications using topics with incompatible mutable types in a very specific situation: if some of the type's members' types differed only in their extensibility kind.

For example, an application writing a topic with type A and an application reading the same topic but using type B should not communicate:

```
// -- Application A: ---------------------------------
struct MemberA {
    string text;
}; //Extensibility MUTABLE_EXTENSIBILITY

struct A {
   MemberA member;
}; //@Extensibility MUTABLE_EXTENSIBILITY

// -- Application B: ---------------------------------
struct MemberB {
    string text;
}; //Extensibility FINAL_EXTENSIBILITY
struct MemberB {
   MemberB member;
}; //@Extensibility MUTABLE_EXTENSIBILITY
```

A and B are incompatible because the extensibility kinds of MemberA and MemberB are different. *Connext* did not detect this incompatibility. This problem has been resolved.

[RTI Issue ID CORE-5720]

3.7.4 Incorrect TypeCode Received for Built-in Types in Some Cases

The received Typecode associated with a remote *DataReader* or *DataWriter* using any of the built-in types (String, KeyedString, Octets, or KeyedOctets) contained an invalid length for the key and value fields if the TypeCode wire representation was disabled by setting **participant_qos.resource_limits.type_code_max_serialized_length** to zero.

Notice that even if the TypeCode wire-representation is disabled, *Connext* still sends type information using TypeObjects (standard type representation in the Extensible Types specification).

[RTI Issue ID CORE-5741]

3.7.5 Possible Segmentation Fault if Two Enumerations were not Assignable

If the types for a *DataWriter* and *DataReader* were not assignable because one of the fields was an enumeration and the enumeration types were not assignable, this may have resulted in a segmentation fault.

In the following example, the enumerator values for the *DataWriter* and *DataReader* have different values and therefore they are not assignable.

DataWriter:

```
enum MyEnum {
    VAL1 = 1,
    VAL2 = 2
};
struct MyType {
    MyEnum m1;
};
```

DataReader:

```
enum MyEnum {
    VAL1 = 3,
    VAL2 = 4
};
struct MyType {
    MyEnum m1;
};
```

This problem has been resolved.

[RTI Issue ID CORE-5808]

3.7.6 No Warning Reported if Two Types not Equivalent when Consistency Kind was DDS_DISALLOW_TYPE_COERCION

Connext failed to log a message indicating that a *DataWriter* and *DataReader* had a type mismatch when the *DataReader's* **type_consistency.kind** was DDS_DISALLOW_TYPE_COERCION.

This problem has been resolved. Now *Connext* will print a warning message:

```
PRESParticipant_compareTypeObjects:types not equivalent: <Type 1 fully
qualified name>, <Type 2 fully qualified name>
```

[RTI Issue ID CORE-5856]

3.7.7 Built-in Type DataReader with type_consistency.kind = DDS_DISALLOW_TYPE_COERCION did not Match with Built-in Type DataWriter using Same Built-in Type

A built-in type *DataReader* with **type_consistency.kind** = DDS_DISALLOW_TYPE_COERCION failed to match with a built-in type *DataWriter*—even if the two types were identical. Consequently, there these entities did not communicate.

Notice that *rtiddspy* sets **type_consistency.kind** = DDS_DISALLOW_TYPE_COERCION in the *DataReaders* created by the utility. Therefore, *rtiddspy's DataReaders* never received data from built-in type *DataWriters.*

[RTI Issue ID CORE-5858]

3.7.8 Non-Readable Format when 'long long' Data Type Sent and Printed with msgTypeSupport_print_data() on Little Endianness Machine

Data of type 'long long' was sent from a *DataWriter* on a big-endian machine to a *DataReader* on a little-endian machine. When the receiving side printed the data with

msgTypeSupport_print_data(), the value was printed in byte format using the native endianness from the sending side (big endian), which is not human readable in little endian machines. This problem has been resolved.

[RTI Issue ID CORE-5879]

3.8 Fixes Related to XML Configuration

3.8.1 Error in Schema for XML Application Description: rti_dds_profiles.xsd

The **rti_dds_profiles.xsd** XSD schema used by the XML-Based Application Creation feature and the *RTI Prototyper* tool has been enhanced to allow the colon ':' to appear in property names.

In addition, an error in the definition of <xs:complexType name="Domain"> has been fixed. This error prevented the XSD from validating with standard XSD-aware tools and prevented autocompletion from working on XML-aware editors.

[RTI Issue ID CORE-5427]

3.8.2 XSD Validation Failed when topic_filter Attribute Contained a Colon

The XSD validation of QoS profiles using the file **rti_dds_qos_profiles.xsd** failed when the attribute **topic_filter** (under the <datawriter_qos>, <datareader_qos>, or <topic_qos> XML tags) contained a value including the colon character ('**:**'). For example:

```
<datawriter_qos topic_filter="MyNS::MyMsg">
```

This problem has been resolved.

[RTI Issue ID CORE-5759]

3.8.3 Possible Memory Leak when Using XML QoS Profile Inheritance

When using XML QoS profile inheritance, there was a chance of a memory leak if the derived profile overrode one of the following values that was also set in the base QoS profile:

- The **output_file** field in the LoggingQosPolicy
- The **name** or **role_name** fields in the EntityNameQosPolicy
- The **flow_controller_name** field in the PublishModeQosPolicy
- The **filter_name** field in the MultiChannelQosPolicy

This problem has been resolved.

[RTI Issue ID CORE-5973]

3.8.4 XML-Based Application Creation Ignored QoS Default Profile when Creating Entities

Entities created using XML-Based Application Creation ignored the default QoS profile if one existed. Internal QoS defaults were used instead. This behavior occurred when no QoS was specified explicitly for the entities defined in the XML file. This problem has been resolved so that in this situation, an existing QoS profile marked as default will be applied.

[RTI Issue ID CORE-5489]

3.9 Fixes Related to Batching

3.9.1 Batch Never Flushed, Even with Finite max_flush_delay

If batching was enabled and the BatchQosPolicy's **max_flush_delay** was set to a finite value, it was possible that a successfully written batch was never automatically flushed (as it should have been based on **max_flush_delay)**. This problem occurred when there was a previously

written batch that was only flushed when the batch that was never flushed was written. This problem has been resolved.

[RTI Issue ID CORE-5870]

3.9.2 wait_for_acknowledgments() Timed Out After Writing Batched Samples

Given a reliable *DataWriter* configured with KEEP_ALL History, enabled batching, and **writer_resource_limits.max_batches** set to a finite value: if the **max_batches** limit was reached, calls to **wait_for_acknowledgments()** may have timed out—even if the *DataWriter* had received all the acknowledgments for successfully flushed batches. This problem has been resolved.

[RTI Issue ID CORE-5949]

3.10 Fixes Related to Request-Reply Communication

3.10.1 C# Requesters/Repliers may have not Communicated with Requesters/Repliers of Other Languages

By default, Requesters and Repliers infer the type name for the Request topic and Reply topic. If the types were inside a module, the type names in the C# API did not exactly match those in the C, C++ or Java APIs, which could cause them not to communicate.

This problem has been resolved; now the C# API generates the same type names as the other APIs.

[RTI Issue ID REQREPLY-11]

3.10.2 Repliers Assigned Incorrect role_name to DataWriters and DataReaders—Java API Only

The *DataWriters* and *DataReaders* created by Requesters and Repliers receive a role_name of "Requester" or "Replier," respectively. The role_name is part of the ENTITY_NAME QosPolicy.

In the previous release, when using the Java API, Repliers mistakenly assigned "Requester" as the role_name for their *DataWriters and DataReaders*. This problem has been resolved; Repliers now assign the correct role name: "Replier."

[RTI Issue ID REQREPLY-11]

3.10.3 Requester and Repliers for DynamicData topics were not supported in C++—AIX platforms only

On AIX platforms, the creation of a Requester or Replier for a DynamicData topic in C++ failed.

For example:

```
Requester<DynamicData, DynamicData> my_dynamic_data_requester(
       RequesterParams(participant)
            .service_name("TestService")
            .request_type_support(my_dynamic_data_support)
            .reply_type_support(my_dynamic_data_support);
```

The constructor failed with a BadParameterException with the following message:

```
connext::details::EntityUntypedImpl::initialize failure caused by
type_support_adapter<DynamicData>::register_type:ERROR: Bad parameter:
DynamicData type support required
```

This problem has been resolved and the constructor no longer fails.

[RTI Issue ID REQREPLY-16]

3.11 Fixes Related to Performance

3.11.1 Extra Traffic Possible when Using Durable Writer History or Persistence Service

Extra traffic may have been generated by a *DataWriter* using Durable Writer History or by a Persistence Service DataWriter. The wire representation of a DATA sample may have contained additional bytes at the end that were not required. This issue affected network efficiency, but did not affect correctness.

[RTI Issue ID CORE-5512]

3.11.2 Subscribing Applications Using Keyed Topics may have Consumed More CPU than Expected

In some scenarios, a subscribing application compiled with *Connext* 5.0.0 exhibited increased CPU usage compared to the same application compiled with *Connext* 4.5f. Release 5.0.0.9 includes several optimizations that significantly reduce or completely eliminate this additional CPU usage.

[RTI Issue ID CORE-5528]

3.11.3 Subscribing Applications Required More CPU

Due to unnecessary lookups in the instance list maintained in a *DataReader* queue, the reception of a new sample required more CPU cycles. This problem has been resolved.

As a workaround, you may have reduced the lookup time by increasing the *DataReader's* QoS value **resource_limits.instance_hash_buckets**. This is no longer necessary.

[RTI Issue ID CORE-5541]

3.11.4 Unexpected Duplicate Samples Received by DataReaders Associated with Durable or Required Subscription

In previous releases, a *DataReader* that was member of a required or durable subscription with quorum *n* may have received a sample even if that sample was already received and acknowledged by *n* different *DataReaders* belonging to the same required or durable subscription.

For required subscriptions, the reception of duplicates occurred when the required subscription's *DataReader* was configured to perform protocol acknowledgement by setting the ReliabilityQosPolicy's **acknowledgment_kind** to DDS_PROTOCOL_ACKNOWLEDGMENT_MODE (the default value).

For durable subscriptions (which require *RTI Persistence Service*), duplicates were always received.

For example, consider the following scenario:

1. Start a *DataWriter*
2. Start *Persistence Service*
3. Create a DurableSubscription with name 'DUR1' and quorum 1
4. Publish same samples
5. Start a *DataReader* for 'DUR1' that receives previous samples
6. Stop the *DataReader* and start a new *DataReader* for 'DUR1'

The second *DataReader* should not receive the samples because they were received by the first *DataReader* on the same DurableSubscription. However, in previous releases, that was not the case. This problem has been resolved.

Note: Even after fixing this issue there are still some conditions under which duplicate samples may be received. For example, in the scenario described above, the second *DataReader* may receive a duplicate if the first *DataReader* crashes or is stopped ungracefully before the *DataWriter* receives the sample acknowledgments.

[RTI Issue ID CORE-5734]

3.12 Fixes Related to Entity Creation and Deletion

3.12.1 Creating/Destroying Multiple Participants in Same Application may have caused Application Crash

A user application that created and destroyed multiple participants may have crashed or reported the following precondition failure when the participant was created/enabled, due to thread ID being reused:

```
REDAWorkerFactory_destroyWorker:!precondition: w->_manager != m
```

This problem has been resolved.

[RTI Issue ID CORE-5657]

3.12.2 Rare Segmentation Fault when Shutting Down Connext Applications—Windows Systems Only

On Windows systems only, in rare cases you may have seen a segmentation fault if the application linked with a dynamic library that uses *Connext* and the dynamic library destroyed a *DomainParticipant* within the **DllMain** entry point before the dynamic library was unloaded. This problem has been resolved.

[RTI Issue ID CORE-5761]

3.12.3 Memory Growth Due to Incomplete Cleanup on Deletion of DataReader or DataWriter

In previous releases, the memory allocated for the "**name**" and "**role_name**" attributes of the DDS_EntityNameQosPolicy was not cleaned up correctly upon deletion of a *DataReader* or *DataWriter*. This issue has now been resolved.

[RTI Issue ID CORE-5331]

3.12.4 DomainParticipantFactory Creation Failed if Monotonic Clock not Available on VxWorks Platforms

In previous releases, DomainparticipantFactory creation could have failed with the following error if Monotonic Clock was not available on VxWorks platforms:

```
RTIMonotonicClock_new:OS clock_getres() failure, error 0X16
DDS_DomainParticipantGlobals_initializeI:!create monotonicClock
DDS_DomainParticipantFactory_newI:!create participant globals
DDS_DomainParticipantFactory_get_instance:!create participant factory
```

This release changes this behavior to only report a warning if Monotonic Clock is not available. The DomainParticipantFactory creation will succeed and *Connext* will use the Real-Time clock instead of the Monotonic Clock.

[RTI Issue ID CORE-4092]

3.12.5 Participant Deletion Triggered Calls to on_data_available() for ParticipantBuiltinTopicDataDataReader

Deleting a Participant triggered calls to a ParticipantBuiltinTopicDataDataReader's **on_data_available()** callback once for each of the matching remote participants. Since deleting a

Participant does not indicate that a matching remote participant has been deleted, the callback was undesirable. This problem has been resolved; deleting a participant no longer triggers the ParticipantBuiltinTopicDataDataReader's **on_data_available()** callback.

[RTI Issue ID CORE-5559]

3.13 Fixes Related to Prototyper (Experimental Feature)

3.13.1 Prototyper Returned Wrong Exit Status

Running *rtiddsprototyper* with the **-help** option resulted in exit status of 1, which incorrectly indicated that an error occurred, even if the execution was successful. This problem has been resolved.

[RTI Issue ID PROT-27]

3.13.2 Prototyper Returned Error if Parameter Contained Spaces

If a parameter to *Prototyper* contained a space (even if it was surrounded by quotes), *Prototyper* reported an error message. For example, this command:

```
$ rtiddsprototyper -cfgFile "File Name.xml"
```

returned an error such as:

```
RTI Connext Prototyper_with_Lua: Unrecognized parameter 'Name.xml'
```

This problem has been resolved.

[RTI Issue ID DIABLO-755]

3.14 Other Fixes

3.14.1 DomainParticipantResourceLimitsQosPolicy Default Values Inconsistent in Java API

Some of the DomainParticipantResourceLimitsQosPolicy default values in Java were inconsistent with the documentation and with the values in the other APIs. They are now consistent with both.

[RTI Issue ID CORE-2698]

3.14.2 Possible Segmentation Fault when property_list_max_length Fields Set to 0

Setting the **participant_property_list_max_length**, **writer_property_list_max_length**, or **reader_property_list_max_length** fields in the DomainParticipantResourceLimitsQosPolicy to 0 caused a segmentation fault in some cases. This problem has been resolved.

[RTI Issue ID CORE-5119]

3.14.3 Find_topic() Crashed with Long Topic Names

Calls to **find_topic()** with a topic name longer than 255 bytes resulted in a segmentation fault. This problem has been resolved.

[RTI Issue ID CORE-5249]

3.14.4 LifespanQosPolicy not Correctly Enforced when Source Timestamp Explicitly Provided with write_w_params or write_w_timestamp

The expiration time of each sample from the *DataWriter's* cache is computed by adding the duration specified by the LifespanQosPolicy to the sample's source timestamp. This expiration time is then compared with the current time obtained from the system clock to identify if a sample has expired.

If the source timestamp provided to **write_w_params()** or **write_w_timestamp()** cannot be compared with the current time obtained using system clock, the LifespanQosPolicy will not correctly applied.

In this release, this behavior has been modified so that the expiration time of each sample in the *DataWriter's* cache is computed by adding the duration specified by the LifespanQosPolicy to the timestamp obtained when creating the sample. Since the creation timestamp and the current time are computed using the same clock, this ensures that the LifespanQosPolicy will be applied correctly.

[RTI Issue ID CORE-5347]

3.14.5 Misleading "invalid designated encapsulation" Log Message

A benign log message, "invalid designated encapsulation," was erroneously printed during normal operation. This problem has been resolved. Now the message will only appear if an invalid CDR encapsulation is encountered.

[RTI Issue ID CORE-5379]

3.14.6 Latency Performance Test for C++: Possible Failure if Max Number Subscriber Set to LATENCY_MAXIMUM_SUBSCRIPTIONS or Higher

This issue pertains to the Latency Performance Test for C++ (provided in **NDDSHOME/example/CPP/performance/latency**). If you invoked the **Latency_publisher** application using LATENCY_MAXIMUM_SUBSCRIPTIONS (32) or higher as the argument for the "subscriber" option, the application may have failed. This problem has been resolved.

[RTI Issue ID CORE-5401]

3.14.7 Setting participant_name More than Once on Disabled Participant Caused Crash

Setting the *DomainParticipant's* EntityNameQosPolicy (**participant_qos.participant_name**) more than once after creating, but before enabling, the *DomainParticipant* may have resulted in a crash. You may have seen an error such as the following:

```
*** glibc detected *** Hello: double free or corruption (out):
0x000000000cff1590 ***
```

This problem has been resolved.

[RTI Issue ID CORE-5409]

3.14.8 ReliableWriterCacheChangedStatus.full_reliable_writer_cache.total_count had Wrong Value

The value of **ReliableWriterCacheChangedStatus.full_reliable_writer_cache.total_count** was incorrectly set to **total_count_change**. Applications would have seen the incremental value of that status since the last time it was looked up, instead of the total value. This problem has been resolved.

[RTI Issue ID CORE-5420]

3.14.9 Error Printing Negative Longs in FooSupport_print_data

The **FooTypeSupport_print_data()** operation did not print negative long values correctly. For example, suppose you had the following IDL type:

```
struct MyType {
    long m1;
};
```

If you set **m1** to -24, **FooTypeSupport_print_data()** printed 4294967272 instead of -24. This problem has been resolved.

[RTI Issue ID CORE-5441]

3.14.10 DataReader May Have Reported Unexpected Loss of Liveliness with DataWriter

A *DataWriter* with Liveliness kind set to DDS_AUTOMATIC_LIVELINESS_QOS may not have sent liveliness messages as often as expected by the *DataReader*. This delay may have caused the *DataReader* to report an unexpected loss of liveliness. This problem has been resolved.

[RTI Issue ID CORE-5442]

3.14.11 Registered Instance Unexpectedly Removed from Keyed DataReader Queue

A registered instance may have been unexpectedly removed from a keyed *DataReader's* queue without transitioning to NOT_ALIVE_NO_WRITERS_STATE. This could have occurred if the **no_writers_generation_count** for the instance was greater or equal to 1 and the *DataReader's* QoS value **reader_resource_limits.max_total_instances** was exceeded. This problem has been resolved.

[RTI Issue ID CORE-5490]

3.14.12 Segmentation Fault or Precondition Error in DataReader when using Exclusive Ownership and filter_redundant_samples is False

In applications using keyed samples, exclusive ownership, and the property **dds.data_reader.state.filter_redundant_samples** set to false, if a *DataWriter* wrote samples to an instance that it did not own, you may have seen a segmentation fault in the *DataReader* when using release libraries, or the following precondition error when using debug libraries:

```
PRESCstReaderCollator_updateLastCommitedSn:!precondition
```

This problem has been resolved.

[RTI Issue ID CORE-5551]

3.14.13 Incorrect Example Files in 'example\JAVA\HelloWorld_xml_compiled'

The example files provided in the directory **example\JAVA\HelloWorld_xml_compiled** were not the correct files for XML-Based Application Creation. This problem has been resolved.

[RTI Issue ID CORE-5555]

3.14.14 Error Parsing Peer Descriptors (initial peers) if Participant ID Limit was an Interval

There was an error when parsing peer descriptors (initial peers) in which the participant ID limit was an interval of the form [low id,high id]. For example, the following UDPv4 descriptor was not parsed correctly:

```
[1,5]@10.10.100.1
```

Connext reported the following error messages when trying to process the above initial peer descriptor:

```
DDS_DiscoveryQosPolicy_parse_peer_descriptor_string:[<low
participant_index_string>, <high participant_index_string>] OR
<participant_index_string> format is unrecognized, string value =
"5]@10.10.100.1"
```

To resolve the problem, the interval separator has changed from a comma (',') to a dash ('-'). For example:

```
[1-5]@10.10.100.1
```

[RTI Issue ID CORE-5615]

3.14.15 Invalid Error Code in Logging Error Messages—Windows Platforms Only

Connext logging messages that contained references to native OS error codes may have reported an invalid native error code of 0 in some scenarios for Windows platforms. This problem has been resolved.

[RTI Issue ID CORE-5617]

3.14.16 Applications Failed to Load Due to Dependency on Floating Point Operation—VxWorks Platforms Only

Connext applications built for VxWorks target platforms may have failed to load. You may have seen this error:

```
undefined symbols: __floatundidf __fixunsdfdi
```

This only occurred if the VxWorks kernel was built without a module to support 64-bit floating point operations. This problem has been resolved.

[RTI Issue ID CORE-5621]

3.14.17 Exceeding Instance Limit did not Trigger DataReader's on_sample_lost() and on_sample_rejected() Callbacks

The on_sample_lost() and on_sample_rejected() callbacks for a DataReader were not triggered when a sample was lost or rejected because of reaching the max_instances resource limit on a DataReader. This problem has been resolved.

[RTI Issue ID CORE-5624]

3.14.18 Keyed DataReader with KEEP_LAST History and not Enough Samples to Satisfy History Depth may have Caused Segmentation Fault

A keyed *DataReader* with a HistoryQosPolicy **kind** of KEEP_LAST and ResourceLimits QoS **max_samples** smaller than the amount [**resource_limits.max_instances** x **history.depth**] may have caused a segmentation fault. This problem only occurred when the queue was full (**max_samples** were in use) and new samples were received. This problem has been resolved.

[RTI Issue ID CORE-5625]

3.14.19 Incompatible Endpoints were Incorrectly Treated as Compatible

A *DataReader*/*DataWriter* pair with the same *Topic* but incompatible QoS policies may have been incorrectly treated as compatible, resulting in the *DataReader* incorrectly receiving samples from the *DataWriter*.

This problem occurred when there were both compatible and incompatible *DataReaders* belonging to the same *DomainParticipant* or when there were both compatible and incompatible *DataWriters* associated with a *DataReader*. This problem has been resolved.

[RTI Issue ID CORE-5639]

3.14.20 Possible Deadlock after Calling ignore_participant() within on_data_available() callback of Participant Built-in Topic DataReader

An application calling **ignore_participant()** within the **on_data_available()** callback of the Participant Built-in Topic DataReader may have caused subsequent calls to *Connext* APIs to lock indefinitely.

This problem only occurred when the user set the Participant Built-in Topic DataReader listener after the *DomainParticipant* was enabled.

[RTI Issue ID CORE-5655]

3.14.21 Calling unregister_thread() from non-User Thread Caused Unpredictable Behavior

In previous releases, if **unregister_thread()** was called in any thread other than a user-created thread, it could result in unpredictable behavior. In this release, calling **unregister_thread()** in any thread other than a user-thread is no longer allowed.

[RTI Issue ID CORE-5658]

3.14.22 Values Set Programmatically for Logging and EntityFactory QoS Policies were Overwritten in Some Cases

Providing values for just the LoggingQosPolicy or just the EntityFactoryQosPolicy via an XML profile resulted in values set programmatically for *both* the LoggingQosPolicy and EntityFactoryQosPolicy to be overwritten.

For example, if only the LoggingQosPolicy was set in an XML profile and the EntityFactoryQosPolicy was modified programmatically, the value for the EntityFactoryQosPolicy was set to the default value. This problem has been resolved. Now if you set only the LoggingQosPolicy in an XML profile, the value set programmatically for the EntityFactoryQosPolicy will not be overwritten.

Note, however, that if the LoggingQosPolicy or EntityFactoryQosPolicy is specified in an XML profile and also set programmatically, the XML profile takes precedence.

[RTI Issue ID CORE-5663]

3.14.23 Calls to DataWriter.get_protocol_status() may have Corrupted Two Constants

Calls to the *DataWriter's* **get_protocol_status()** operation may have corrupted the values for the constants InstanceHandle_t.HANDLE_NIL and SequenceNumber_t.SEQUENCE_NUMBER_UNKNOWN. Any subsequent operations receiving or using these constants (for example, calling **write(data, InstanceHandle_t.HANDLE_NIL))** may have failed. This problem has been resolved.

[RTI Issue ID CORE-5664]

3.14.24 Inconsistent QoS Properties in PublishModeQosPolicy and TypeConsistencyEnforcementQosPolicy were Reported as Warnings, Now are Exceptions

Inconsistent values specified for the PublishModeQosPolicy and TypeConsistencyEnforcementQosPolicy were previously reported as warnings. Now they will be reported as exceptions.

[RTI Issue ID CORE-5722]

3.14.25 Samples Unexpectedly Removed from Durable Writer History after Restore Operation

If a *DataWriter* using Durable Writer History was restarted after a graceful or ungraceful shutdown, the samples in the *DataWriter's* history were removed unexpectedly if the following conditions were met:

- The *DataWriter* was configured with a finite lifespan
- The *DataWriter's* property **dds.data_writer.history.odbc_plugin.in_memory_state** was set to zero

This issue affected both *DataWriters* using Durable Writer History and *RTI Persistence Service*.

[RTI Issue ID CORE-5752]

3.14.26 Sequences in Java not Properly Resized

Some Java sequences were not being properly shortened and always retained one element more than they should have. This behavior would have allowed referencing data that should have been removed. This is no longer the case.

[RTI Issue ID CORE-5762]

3.14.27 Possible Segmentation Fault when Type Objects Received in 64-bit Windows Connext Applications

In previous releases, 64-bit Windows *Connext* applications may have issued a segmentation fault when type objects were received. The workaround was to disable type-object deserialization by setting **participant_qos.resource_limits.type_object_max_deserialized_length** and **participant_qos.resource_limits.deserialized_type_object_dynamic_allocation_threshold** to zero. This problem has been resolved.

[RTI Issue ID CORE-5789]

3.14.28 Retrieving Protocol Status from Disabled DataReader Could Cause Crash

Attempting to retrieve the protocol status from a disabled *DataReader* created from an enabled *DomainParticipant* may have resulted in a segmentation fault. This is a valid operation and the problem has been resolved.

[RTI Issue ID CORE-5806]

3.14.29 DataReaders of Unkeyed Topics Incorrectly Reported Missed Deadlines if No Matching DataWriters

A *DataReader* calls **on_requested_deadline_missed()** every time its requested-deadline period in the DeadlineQosPolicy elapses without receiving new data samples. But when there are no *DataWriters* communicating with that *DataReader*, the middleware should not consider that the deadline has been missed. *DataReaders* of unkeyed topics still incorrectly called **on_requested_deadline_missed()** in this situation.

This problem has been resolved; now *DataReaders* will report a missed deadline only if the requested period elapses while there is at least one matching *DataWriter*.

[RTI Issue ID CORE-5860]

3.14.30 Potential Deadlock when using set_qos_with_profile() to set QoS for Participant Configured to use Monitoring Library

It was possible to run into a deadlock situation when trying to set QoS for a *DomainParticipant* using the **set_qos_with_profile()** operation. The deadlock was only possible if the *DomainParticipant* had also been configured to use monitoring libraries. This problem has been resolved.

[RTI Issue ID CORE-5859]

3.14.31 Precondition Error if DataReader had AUTO or EXPLICIT acknowledgment_kind and Matching DataWriter sent Virtual HeartBeats

When using the debug libraries, *Connext* logged a precondition error if a *DataReader* configured with the ReliabilityQosPolicy's **acknowledgment_kind** set to AUTO or EXPLICIT matched a *DataWriter* configured to send virtual HeartBeats (by adjusting the DataWriterProtocolQosPolicy's **rtps_reliable_writer.virtual_heartbeat_period** and/or **rtps_reliable_writer.samples_per_virtual_heartbeat**).

The precondition error was:

```
PRESCstReaderCollator_commitVirtualWriter:!precondition
```

This was a benign message that did not affect correctness of program execution.

This problem has been resolved, the message will no longer be logged in this situation.

[RTI Issue ID CORE-5865]

3.14.32 Source Timestamp Mismatch when using write_w_timestamp()

In previous releases, the source timestamp associated with a sample published with the **write_w_timestamp()** operation may have been different than the source timestamp associated with the same sample when it was received by a *DataReader*. When different, the timestamp was off by one nanosecond. This problem has been resolved.

[RTI Issue ID CORE-5926]

3.14.33 Wrong Behavior when Using Partition in which Last Element was Empty String

When using a partition in which the last element of the sequence of names was an empty string, you may have seen the following error:

```
DDS_StringSeq_get_reference:!assert index out of bounds
```

Such a partition was still applied but its retrieval via **get_qos()** was missing the last element. This problem has been resolved; now the empty string partition is always present in the sequence of partitions, regardless of the position of the sequence in which it appears.

[RTI Issue ID CORE-5998]

3.14.34 wait_for_historical_data Returned Immediately if Previous Call Timed Out

If a call to **wait_for_historical_data()** has timed out, it was possible that the next call to **wait_for_historical_data()** would return immediately with an OK return code, even if there was historical data still to be received. This problem has been resolved.

[RTI Issue ID CORE-6012]

3.14.35 Potential Deadlock when using Some APIs

There was potential for a deadlock risk when using the following APIs:

- *DomainParticipant's* **delete_contentfilteredtopic()**
- *DomainParticipant's* **get_discovered_participant_data()**
- *DomainParticipant's* **get_discovered_topic_data()**
- *DataReader's* **add_remote_writer_queue()**
- *DataWriter's* **flush()**

When the problem occurred, there would be an error message similar to the following:

```
[D0075|Reader(8000003D)|T=DCPSSubscription|GET_MATCHED Participant
DATA|D0075|GET_DISCOVERED Participant
DATA]REDAWorker_enterExclusiveArea:worker rR010751e00b50 deadlock risk:
cannot enter 200e3118 of level 40 from level 10
```

This problem has been resolved.

[RTI Issue ID CORE-6019]

3.14.36 Some Samples not Delivered to Application when using a 'take' Operation

In previous releases some samples may not have been delivered to the application after using the **take()**, **take_instance()**, **take_next_sample()**, or **take_next_instance()** operations.

This problem only occurred when *all* the following conditions were satisfied:

- ❏ Topic was keyed
- ❏ PresentationQosPolicy's **access_scope** was DDS_INSTANCE_PRESENTATION_QOS or PresentationQosPolicy's **ordered_access** was TRUE
- ❏ If the **take()** or **take_instance()** APIs were used, the **sample_state** parameter was DDS_NOT_READ_SAMPLE_STATE

This problem has been resolved.

[RTI Issue ID CORE-6023]

3.14.37 "Indicator variable required but not supplied" Error when using Required Subscriptions or Application Acknowledgment with Durable Writer History

When using required subscriptions or application-level acknowledgment with durable writer history, you may have seen an "Indicator variable required but not supplied" error. This error only occurred when using MySQL as the external database.

```
!fetch virtual writer info - ODBC: error: 22002 0 [unixODBC][MySQL][ODBC
5.1 Driver][mysqld-5.1.44-community]Indicator variable required but not
supplied
```

This problem has been resolved.

[RTI Issue ID CORE-6036]

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 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]

4.7 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.

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

4.8 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]

4.9 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]

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]

- Types that contain bit fields are not supported by DynamicData. Therefore, when *rtiddsspy* discovers any type that contains a bit field, *rtiddsspy* will print this message:

  ```
  DDS_DynamicDataTypeSupport_initialize:type not supported (bitfield member)
  ```

  [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]

4.13 Typecodes Required for Request-Reply Communication Pattern

Typecodes are required when using the Request-Reply communication pattern[1]. To use this pattern, do not use *rtiddsgen*'s **-noTypeCode** flag. If typecodes are missing, the Requester will log an exception.

[RTI Issue ID REQREPLY-3]

1. The Request-Reply communication pattern is only available with *Connext Messaging*, not with *Connext DDS*. (If you have *Connext Professional Edition*, you have *Connext Messaging*.)

4.14 To Declare Arrays as Optional in C/C++, They Must be Aliased

When generating C or C++ code, arrays cannot be declared as optional unless they are aliased.

[RTI Issue ID CODEGEN-604]

4.15 Code Generator Unable to Detect if Optional Member is Inside Aggregated Key Member

The *rtiddsgen* code generator cannot detect if an optional member is inside an aggregated key member.

[RTI Issue ID CODEGEN-605]

4.16 Classes and Types Defined in Some .NET Namespaces cannot be used to Define User Data Types

The name of the classes and types defined in the following .NET namespaces cannot be used to define user data types:

- System
- System::Collections
- DDS

For example, if you try to define the following enumeration in IDL:

```
enum StatusKind{
    TSK_Unknown,
    TSK_Auto
};
```

The compilation of the generated CPP/CLI code will fail with the following error message:

```
error C2872: 'StatusKind' : ambiguous symbol
```

The reason for this error message is that the enumeration StatusKind is also defined in the DDS namespace and the generated code includes this namespace using the "using" directive:

```
using namespace DDS;
```

The rational behind using the "using" directive was to make the generated code shorter and more readable.

[RTI Issue ID CODEGEN-547]

4.17 Possible Race Condition during Participant Deletion may Produce "deadlock risk" Error

In limited scenarios, deletion of *DomainParticipants* may produce a "deadlock risk" error message and prevent completion of DDS operations being executed in other threads. This issue only occurs in applications that have created multiple *DomainParticipants*. The specific error message will be similar to:

```
REDAWorker_enterExclusiveArea:worker U70f73cc0 deadlock risk: cannot enter
0x10182a180 of level 30 from level 50
```

In many cases, this issue can be remedied by deleting *DomainParticipants* upon exiting or when no other DDS operations are performed.

[RTI Issue ID CORE-6045]

5 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 HTML documentation, experimental APIs are marked with **<<experimental>>**.

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

Disclaimers

- 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/).