RTI Data Distribution Service

Core Libraries and Utilities

Release Notes

Version 4.5e

The Global Leader in DDS

© 2011 Real-Time Innovations, Inc.
All rights reserved.
Printed in U.S.A. First printing.
October 2011.

Trademarks

Real-Time Innovations and RTI are registered trademarks of Real-Time Innovations, Inc.
All other trademarks used in this document are the property of their respective owners.

Copyright Restrictions

No part of this publication may be reproduced, stored in a retrieval system, or transmitted in any form (including electronic, mechanical, photocopy, and facsimile) without the prior written permission of Real-Time Innovations, Inc. The software described in this document is furnished under and subject to the RTI software license agreement. The software may be used or copied only under the terms of the license agreement.

Technical Support

Real-Time Innovations, Inc.
232 East 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 6.5e (Section 3)
- ❑ Known Issues (Section 4)
- ❑ Custom Supported Platforms (Section 5)

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

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

1 System Requirements

1.1 Supported Operating Systems

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

In this context, a *host* is the computer on which you will be developing a *RTI Data Distribution Service* 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 an *RTI Data Distribution Service* application for any architecture. You will also need a target installation, which provides the libraries required to build an *RTI Data Distribution Service* application for that particular target architecture.

Table 1.0 lists the platforms available with *RTI Data Distribution Service* 4.5e.

Table 1.0 **Platforms Available with Release 4.5e**

Platform	Operating System	Reference
AIX5	AIX 5.3	Table 1.1 on page 3
DEFAULTRTI	DEFAULTRTI 5.0.11, 10	Table 1.2 on page 5
Linux2.6 (CLARE™)	Linux kernel 2.6.32	Table 1.3 on page 9
Linux (Intel32)	Linux RHEL 4.5	Table 1.4 on page 9
	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, 5.3, 5.5, 6.0, 6.1	
	Red Hat Enterprise Linux 5.3 with Real-Time Extensions	
	SUSE® Linux Enterprise Server 10.1 (2.6 kernel)	
	Ubuntu® Server 10.1 (2.6 kernel)	
Linux (PowerPC®)	Freescale™ P2020RDB (2.6.32 kernel)	
	RH Linux (2.4.20 kernel)	Table 1.5 on page 11
	Wind River® Linux 3	
	Yellow Dog™ Linux 4.0	
LynxOS®	LynxOS 4.0, 4.2, 5.0	Table 1.6 on page 11
Mac OS®	Mac OS X 10.6	Table 1.7 on page 12
QNX®	QNX Neutrino®6.4.1, 6.5	Table 1.8 on page 12

Table 1-2 **Platforms Available with Release 4.5e**

Platform	Operating System	Reference
SolarisOS	Solaris 10, 11	Table 1-3 on page 1-4
VxWorksE	VxWorks 5.5, 6.2 - 6.9	
VxWorks x86	VxWorks x86 7.2	Table 1-11 on page 1-9
VxWorks MIL-O	VxWorks MIL-O 1.1	
Windows®	Windows 7	
	Windows 2008 with service pack 2 or higher	
	Windows 2012 and Windows 2016 x64 Edition	
	Windows Vista®	Table 1-11 on page 1-9
	Windows Server 2008 R2	
	Windows XP Professional	
	Windows XP Professional x64 Edition	

Optional Requirement for Release 4.5e

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 4.5e dll. Versions of the following RTLs include packages:
  - msvcm32d.dll and msvcm40d.dll
  - msvcr32d.dll and msvcr40d.dll
  - msvcr32d.dll and msvcrt40d.dll

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=26367

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:
  - *librtiVx2VS2008* built with dynamic libraries
  - *winWinnetVS2008* 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 architecture:
  http://www.microsoft.com/downloads/details.aspx?displaylang=en&FamilyID=A5C36275-38F7-4AB7-9AED-38022EAFFC2a&displaylang=en
- For x64 architecture:
  http://www.microsoft.com/downloads/details.aspx?displaylang=en&FamilyID=bd7a2a7e-327f-4b46-b2c-57f2ebc6a&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:
  - *librtiVx2VS2010* built with dynamic libraries
  - *winWinnetVS2010* built with dynamic libraries
  - *librtiVxExtensionHandsWinnetExtensionHandsVS2010*

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 architecture:
  http://www.microsoft.com/download/en/details.aspx?id=5555
- For x64 architecture:
  http://www.microsoft.com/download/en/details.aspx?id=14632

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

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

- Table 1.1, "AIX Platforms," on page 1-9
- Table 1.2, "INTEGRITY Platforms," on page 1-9
- Table 1.3, "Linux Platforms on Cell BE CPUs," on page 1-9
- Table 1.4, "Linux Platforms on Intel and AMD CPUs," on page 1-9
- Table 1.5, "Linux Platforms on PowerPC CPUs," on page 1-11
- Table 1.6, "SymOS Platforms," on page 1-11
- Table 1.7, "Mac OS Platforms," on page 1-11
- Table 1.8, "QNX Platforms," on page 1-12
- Table 1.9, "Solaris Platforms," on page 1-12
- Table 1.10, "Supported VxWorks Target Platforms," on page 1-12
- Table 1.11, "Windows Platforms," on page 1-12

1.2 Platforms Removed from RTI Data Distribution Service 4.5e

Beginning with 4.5e, these platforms are no longer supported:

Operating System		CPU	Compiler	RTI Architecture Abbreviation
INTEGRITY	INTEGRITY v.4.0	PPC 7400	IsoTerrible (GHeavy) TCP/IP stack	ppc7400eabihf-32-7.xenomai50-DT-debs
	INTEGRITY v.4.0	PPC 7400	isoterrmpx TCP/IP stack (with multicast)	ppc7400eabihf-32-7.xenomai50-DT-debs-pk
	INTEGRITY v.5.0	PPC 7400	IsoTerrible (GHeavy) TCP/IP stack	ppc7400eabihf-32-7.xenomai50-DT-debs
	INTEGRITY v.5.0 RH-10	PPC 7400	GHeavy TCP/IP stack	ppc7400eabihf-32-7.xenomai50-DT-debs-gheavy-rh

Operating System		CPU	Compiler	RTI Architecture Abbreviation
Linux	Fedora 9 (2.6 kernel)	64-bit x86	gcc 4.3.1	x64Linux2.4gcc4.1
			Sun Java Platform Standard Edition JDK 1.5 and 1.6	x64Linux2.4sun1.4gcc4.1.1jdk
	Fedora 10 (2.6 kernel)	x86	gcc 4.3.2	x86Linux2.6gcc4.3.2
			Sun Java Platform Standard Edition JDK 1.6	x86Linux2.6gcc4.3.2jdk
	Mandriva (2.6.20 kernel)	Power/PCI-EXPRESS	gcc 4.1.0 with GNU libc 2.6	ppcLinux2PCIexpress2.4gcc4.1.0
	Red Hat Enterprise Linux 5 (2.6 kernel)	Pentium-class	gcc 3.2.3	i86Linux2.4gcc3.2
			Sun Java Platform Standard Edition JDK 1.5 and 1.6	i86Linux2.4gcc3.2jdk
		AMD64	gcc 3.2.3	x64Linux2.4gcc3.2
			Sun Java Platform Standard Edition JDK 1.5 and 1.6	x64Linux2.4gcc3.2jdk
	Red Hat Linux, RH (2.4 kernel)	Pentium-class	gcc 3.2.3	i86Linux2.4gcc3.2
			Sun Java Platform Standard Edition JDK 1.5 and 1.6	i86Linux2.4gcc3.2jdk
	Red Hat Linux RH (2.6 kernel)	Pentium-class	gcc 3.2.3	i86Linux2.6gcc3.2
			Sun Java Platform Standard Edition JDK 1.5 and 1.6	i86Linux2.6gcc3.2jdk
LynxOS	LynxOS 5.0	Pentium-class	gcc 3.2.2	i86LynxOS MCH3.0gcc3.2.2
	LynxOS 4.0	PPC 970	gcc 3.2.2	ppc7400LynxOS4.0gcc3.2.2
		PPC 7xx (such as 750)	Sun Java Platform Standard Edition JDK 1.4	ppc7400LynxOS4.0gcc3.2.2jdk
	LynxOS 3.2	Pentium-class	gcc 2.7.2	i86LynxOS3.2gcc2.7
Solaris	Solaris 2.9	UltraSPARC	gcc 3.2.7	sparcSol2.9gcc3.2
	Solaris 2.10	UltraSPARC (with native 32-bit support)	cc 5.9	sparcSol9x32 Sun5.9

Operating System		CPU	Compiler	RTI Architecture Abbreviation
VxWorks	VxWorks 5.4.x	ppc603, ppc604, ppc750, ppc7400	gcc-2.96	ppc603/VxSupe
	VxWorks 5.5.1	ppc405	gcc-2.96	ppc405/VxSupe
	VxWorks 6.0	Pentium	gcc-2.96	pentium/VxSupe
	VxWorks 6.0, 6.1	Pentium	gcc-3.3.2	pentium/VxSupe/3.3.2 pentium/VxSupe/3.3.2 For Real Time Processors: pentium/VxSupe/3.3.2_rtp
	VxWorks 6.0, 6.1 Sol	Sun Ultra2 CPU running Solaris hardware	Sun Ultra2 PPC/32 mlib gcc-3.3.2	For kernel modules: pentium/VxSupe/3.3.2 For Real Time Processors: ppc603/VxSupe/3.3.2_rtp
	VxWorks 6.5, 6.6	Pentium	gcc-3.4.4	For kernel modules: pentium/VxSupe/3.4.4 For Real Time Processors: pentium/VxSupe/3.4.4_rtp
	VxWorks 6.9	Pentium	gcc-3.4.4	For kernel modules: pentium/VxSupe/3.4.4 For Real Time Processors: pentium/VxSupe/3.4.4_rtp
Windows	Windows 2000	x86	Visual C 6.0	dotNet/32/C/D
			Visual Studio 2005	dotNet/32/C/D
			Visual Studio 2010	dotNet/32/C/D
	Windows 2003	x86	Visual C 6.0	dotNet/32/C/D
			Visual C 7.0	dotNet/32/C/D
			Visual Studio 2005	dotNet/32/C/D/I/S
			Visual Studio 2005SP1 if C++ 8.0	amwr/dNetC.lib /I/S/D/I/S
	Windows CE 6.0	arm-4	Visual Studio 2005SP1 if C++ 8.0	dotNet/arm/CE/I/S/D/I/S
	Windows XP (Professional)	x86	Visual C 6.0	dotNet/32/C/D
			Visual C 7.0	dotNet/32/C/D
			Visual Studio 2005	dotNet/32/C/D/I/S

1.3 Disk and Memory Usage

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

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

1.4 Networking Support

RTI Data Distribution Service includes full support for pluggable transports. *RTI Data Distribution Service* applications can run over various communication media, such as UDP/IP or shared memory. Applications can use shared memory—provided the correct "transport plug-in" for the media is installed.

By defaulty, *RTI Data Distribution Service* uses the UDP/IPv4 and shared-memory transport plug-in. The shared memory transports not supported for VxWorks 5.5.

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

- Linux / Fedora: all platforms except SELinux (2x.32 kernel), Freescale P2020RDB (2x.32 kernel), and Wind River Linux 3
- Solaris: all platforms
- VxWorks x7 and 6.8 (gcc600/x7gcc4.12)
- Windows: all platforms.

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

- Red Hat Enterprise Linux 4.0, 5.0-5.2, 5.4, 5.5, 6.1; CentOS 5.4, 5.6, and Ubuntu 10.04
- Windows x-bit Visual Studio 2005, 2008, and 2010.

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

Table 1.1 **AIX Platforms**

Operating System	CPU	Compiler	RTI Architecture Abbreviation
AIX 5.3	POWER5 (32-bit mode)	IBM XLC for AIX v9.0	aix5.3/ppc32
		IBM Java 1.4	aix5.3/ppc32j14
	POWER5 (64-bit mode)	IBM XLC for AIX v9.0	aix5.3/ppc64-9
		IBM Java 1.4	aix5.3/ppc64j14

Table 1.2 **INTEGRITY Platforms**

Operating System	CPU	Compiler	IP Stack	RTI Architecture Abbreviation
INTEGRITY 5.0.10	PPC 90xx	multi 4.1.4	4.3b-mt IP stack	ppc90x-integrity-10.11.ucm-p2020
INTEGRITY 10.0.0	x86	multi 5.0x	4.3.0 beta IPv4 stack	pentium-integrity-10.0.0-pentiu

1. Kernel must be built using *Ipd.ucr -Ipd*.

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

Operating System	CPU	Compiler	RTI Architecture Abbreviation
Linux (2.6.18 kernel)	Cell/BE	gcc 4.0.1, gdb 7.0	cellbelinux-2.6.gcc4.0.1

Table 1.4 **Linux Platforms on Intel and AMD CPUs**

Operating System	CPU	Compiler	RTI Architecture Abbreviation
CentOS 5.4, 5.5 (2.6 kernel)	x86	gcc 4.1.2	x86Linux2.6gcc4.1.2
		Sun Java Platform Standard Edition JDK 1.6	x86Linux2.6gcc4.1.2jdk
	x64	gcc 4.1.2	x64Linux2.6gcc4.1.2
		Sun Java Platform Standard Edition JDK 1.6	x64Linux2.6gcc4.1.2jdk
Fedora 12 (2.6 kernel)	x64	gcc 4.4.4	x64Linux2.6gcc4.4.4
Fedora 12 (2.6.32 kernel) with gcc 4.3.3	x86	gcc 4.3.3	x86Linux2.6gcc4.3.3
		gcc 3.4.3	x86Linux2.6gcc3.4.3
Red Hat Enterprise Linux 4.8 (2.6 kernel)	x86	Sun Java Platform Standard Edition JDK 1.5 and 1.6	x86Linux2.6gcc3.4.3jdk
		gcc 3.4.3	x86Linux2.6gcc3.4.3
	x64	Sun Java Platform Standard Edition JDK 1.5 and 1.6	x64Linux2.6gcc3.4.3jdk
Red Hat Enterprise Linux 5.4 (2.6 kernel)	x86	gcc 4.1.1	x86Linux2.6gcc4.1.1
		Sun Java Platform Standard Edition JDK 1.5 and 1.6	x86Linux2.6gcc4.1.1jdk
	x64	gcc 4.1.1	x64Linux2.6gcc4.1.1
		Sun Java Platform Standard Edition JDK 1.5 and 1.6	x64Linux2.6gcc4.1.1jdk

Table 1-4 **Linux Platforms on Intel and AMD CPUs**

Op-ating System	CPU	Compiler	ICTI Architecture Abbreviation
Red Hat Enterprise Linux 9.1, 7.2, 8.6, 5.10 (Linux-64)	x86	gcc 4.2.2	x86linux32.dgcc4.2.2
		Sun Java Platform Standard Edition JDK 1.6	x86linux32.dgcc4.1.2.dpkb
	x64	gcc 4.2.2	x86linux64.dgcc4.1.2
		Sun Java Platform Standard Edition JDK 1.6	x86linux64.dgcc4.1.2.dpkb
Red Hat Enterprise Linux 9.2 with Real-Time Extensions (Linux-64)	x86	gcc 4.2.2	x86linux32.dgcc4.1.2
		Sun Java Platform Standard Edition JDK 1.5 and 1.6	x86linux32.dgcc4.1.2.dpkb
Red Hat Enterprise Linux 6.0, 6.1 (Linux-64)	x86	gcc 4.4.5	x86linux32.dgcc4.4.5
		gcc 4.4.5	x86linux32.dgcc4.4.5
SUSE Linux Enterprise Server 10.1 (Linux-64)	x86	gcc 4.1.0	x86linux32.dgcc4.1.0
		Sun Java Platform Standard Edition JDK 1.5 and 1.6	x86linux32.dgcc4.1.0.dpkb
	x64	gcc 4.1.0	x86linux64.dgcc4.1.0
		Sun Java Platform Standard Edition JDK 1.5 and 1.6	x86linux64.dgcc4.1.0.dpkb
Ubuntu Server 10.04 (Linux-64)	x86	gcc 4.4.3	x86linux32.dgcc4.4.3
		Sun Java Platform Standard Edition JDK 1.5 and 1.6	x86linux32.dgcc4.4.3.dpkb
	x64	gcc 4.4.3	x86linux64.dgcc4.4.3
		Sun Java Platform Standard Edition JDK 1.5 and 1.6	x86linux64.dgcc4.4.3.dpkb

Table 1.6 **Linux Platforms on PowerPC CPUs**

Operating System	CPU	Compiler	RT1 Architecture Abbreviation
RH Linux (2.4.21 kernel)	PowerPC 7400SP	gcc 4.0.1, gfdec 4.0	ppc64-rel-PPLinux-2.6gcc-4.3.1
SuSEnx Dsp32 Linux 8.0 (2.4.kernel) (target only)	EPIC 7Mx (with no 7400)	gcc 3.3.3	ppc7400LinuxG2gcc3.3.3
Fedora-dw PXD2000B (2.6.7 kernel)	EPIC 9Mx	Fedora-dw gcc-3.3.76 based on gcc-4.0.2	ppc9Mx-dLinux-2.6gcc-4.0.2
Wind River Linux 3	EPIC 9Mx	gcc 4.3.2	ppc9MxGNULinux-2.6gcc-4.3.2

Table 1.6 **LynxOS-PowerPC**

Operating System	CPU	Compiler	RT1 Architecture Abbreviation
LynxOS 4.0	n/a	gcc 3.2.2	ppc4Lynx-4.0gcc-3.2
		Sun Java Platform Standard Edition SDK 1.4	ppc4Lynx-4.0gcc-3.2-Jpdk
	EPIC 7Mx (with no 7400)	gcc 3.2.2	ppc7400Lynx-4.0gcc-3.2.2
		Sun Java Platform Standard Edition SDK 1.4	ppc7400Lynx-4.0gcc-3.2-Jpdk
LynxOS 4.2	EPIC 7Mx (with no 7400)	gcc 3.2.2	ppc7400Lynx-4.2.0gcc-3.2.2
LynxOS 5.0	EPIC 7Mx (with no 7400)	gcc 3.4.3	ppc7400Lynx-5.0.0gcc-3.4.3
		Sun Java Platform Standard Edition SDK 1.4	ppc7400Lynx-5.0.0gcc-3.4.3-Jpdk

Table 1.7 **Mac OS Platforms**

Operating System	CPU	Compiler	RT1 Architecture Abbreviation
Mac OS X	n/a	gcc-4.2.1	x86DarwinGcc-4.2.1
		Java SDK 1.6 for Mac OS	x86DarwinGcc-4.2.1-Jdk

Table 1-8 **GNX Platforms**

Operating System	CPU	Compiler	RTI Architecture Abbreviation
GNX Neutrino 6.4.1	x86	qcc 4.3.3 with GNU C++ libraries	i86QNX6.4.3qcc_gpp
GNX Neutrino 6.5	x86	qcc 4.4.2 with GNU C++ libraries	i86QNX6.5qcc_gpp4.4.2

Table 1-9 **Solaris Platforms**

Operating System	CPU	Compiler	RTI Architecture Abbreviation
Solaris 2.9	x86	gcc 3.3.2	i86sol2.9gcc3.3.2
		Sun Java Platform Standard Edition JDK 1.5 and 1.6	i86sol2.9jdk
		C++ 5.4 (Forte Dev 7, Sun One Studio 7)	sparcv9sol2.9cc5.4
	UltraSPARC	gcc 3.3	sparcv9sol2.9gcc3.3
		Sun Java Platform Standard Edition JDK 1.5 and 1.6	sparcv9sol2.9jdk
Solaris 2.10	AMD64	gcc 4.4.3	amd64sol2.10gcc4.4.3
		Sun Java Platform Standard Edition JDK 1.5 or 1.6	amd64sol2.10jdk
	x86	gcc 4.4.2	i86sol2.10gcc4.4.2
		Sun Java Platform Standard Edition JDK 1.5 or 1.6	i86sol2.10jdk
	UltraSPARC	gcc 3.4.2	sparcv9sol2.10gcc3.4.2
		Sun Java Platform Standard Edition JDK 1.5 or 1.6	sparcv9sol2.10jdk
	Ultra-SPARC-IIIi-based or IIII-based	gcc 3.4.2	sparcv9sol2.10gcc3.4.2
		Sun Java Platform Standard Edition JDK 1.5 or 1.6	sparcv9sol2.10jdk

Table 1-10 **Supported VxWorks Target Platforms[1]**

Operating System	CPU	Compiler	RTI Architecture
VxWorks 5.5	PPC 603	gcc 2.96	ppcnf603vx5.5gcc
	PPC 604	gcc 2.96	ppcnf604vx5.5gcc
	PPC 750	gcc 2.96	ppcnf750vx5.5gcc
	PPC 7400	gcc 2.96	ppcnf7400vx5.5gcc

Table 1-12: **Supported VxWorks Target Platforms**[1]

Operating System	CPU	Compiler	RTI Architecture
VxWorks 5.5, 6.4	Any Wind River PPC32 CPU with floating point hardware	gcc 3.4.4	For kernel modules: ppc603vx.3gcc3.4.4 For Real Time Processes: ppc603vx.3gcc3.4.4.zip
VxWorks 6.5	Any Wind River PPC32 CPU with floating point hardware	gcc 3.4.4	For Kernel Modules: ppc603vx.3gcc3.4.4 For Real Time Processes: ppc603vx.3gcc3.4.4.zip
VxWorks 6.6	Freescale	gcc 4.1.2	For Kernel Modules: powerpcnetCoreVxpcs4.1.2 For Real Time Processes: powerpcnetCoreVxpcs4.1.2.zip
	Any Wind River PPC32 CPU with floating point hardware	gcc 4.1.2	For Kernel Modules: ppc603vx.3gcc4.1.2 For Real Time Processes: ppc603vx.3gcc4.1.2.zip
	PPC 405[2]	gcc 4.1.2	For Kernel Modules: ppc405vx.3gcc4.1.2 For Real Time Processes: ppc405vx.3gcc4.1.2.zip
VxWorks 6.7	Freescale	gcc 4.1.2	For Kernel Modules: powerpcnetCoreVxpcs4.1.2 For Real Time Processes: powerpcnetCoreVxpcs4.1.2.zip
	Any Wind River PPC32 CPU with floating point hardware	gcc 4.1.2	For Kernel Modules: ppc603vx.3gcc4.1.2 For Real Time Processes: ppc603vx.3gcc4.1.2.zip For non-SMP systems: ppc603vx.3gcc4.1.2 For SMP systems: ppc603vx.3gcc4.1.2_smp
	PPC 405[2]	gcc 4.1.2	For Kernel Modules: ppc405vx.3gcc4.1.2 For Real Time Processes: ppc405vx.3gcc4.1.2.zip

Table 1-10: **Supported VxWorks Target Platforms**[1]

Operating System	CPU	Compiler	RTI Architecture
VxWorks 6.8	Pentium	gcc 4.1.2	Pre-Kernel Modules: pentium/vxt.3gcc4.1.2 Pre Real Time Processes: pentium/vxtd.gcc4.1.2_rtp
	Any Wind River PPC32 CPU with floating point hardware	gcc 4.1.2	Pre-Kernel Modules: ppc603vxt.3gcc4.1.2 Pre Real Time Processes: no a.out/ELF system: ppc603vxt.3gcc4.1.2_rtp
VxWorks 6.9	Pentium32-bit	gcc 4.3.3	Pre-Kernel Modules: pentium/vxt5.3gcc4.3.3 Pre Real Time Processes: pentium/vxt5.3gcc4.3.3_rtp
	Pentium 64-bit	gcc 4.3.3	Pre-Kernel Modules: pentium64/vxt5.3gcc4.3.3 Pre Real Time Processes: pentium64/vxt5.3gcc4.3.3_rtp
	Any Wind River PPC32 CPU with floating point hardware	gcc 4.3.3	Pre-Kernel Modules: ppc603vxt5.3gcc4.3.3 Pre Real Time Processes: ppc603vxt5.3gcc4.3.3_rtp
VxWorks 653 2.3	x86-64vxt	gcc 3.2	x86-64vxt.Vxx653.2.3gcc3.3.2
	PMPC	gcc 3.2	ppcmpc5xx653.2.3gcc3.3.2
VxWorks MILS 2.1.2 with = Threads 2.1.2 (VxWorks 5.5.1)	ppc604	gcc 3.3.2	ppc604Vx.V2.2.3gcc3.3.2

Legend GN:

1. For use with Windows and for Solaris Hosts as supported by Wind River Systems.
2. For ppc603, the architecture string is the same for VxWorks 6.8 and 6.7.

Table 1.1.1 **Windows Platforms**

Operating System	CPU	Compiler or Software Development Kit [1] [2]	RT1 Architecture
Windows 7	x86	Visual Studio 2010	0WINDOWS1010
		Visual Studio 2008 (C++ /CLI, C# 2.0 or 3.0)	0WINDOWS2005v8.0
		Sun Java Platform Standard Edition JDK 1.5 or 1.6	0WINDOWS1500
Windows 7 x64 Edition	x64	Visual Studio 2010	x64WINDOWS1010
		Visual Studio 2008 (C++ /CLI, C# 2.0 or 3.0)	x64WINDOWS2005v8.0
		Sun Java Platform Standard Edition JDK 1.5 or 1.6	x64Windows
Windows 2000	x86	Visual Studio 2005 SP1	0WINDOWS2005
		Visual Studio 2008 SP 1 (C++ /CLI, C# 2.0 or 3.0)	0WINDOWS2005v8.0
		Sun Java Platform Standard Edition JDK 1.5 or 1.6	0WINDOWS1500
Windows 2003	x86	Visual Studio 2008 SP 1	0WINDOWS2005v8.0
		Visual Studio 2005 SP 1 (C++ /CLI, C# 2.0 or 3.0)	0WINDOWS2005
		Visual Studio 2008 SP 1 (C++ /CLI, C# 2.0 or 3.0)	0WINDOWS2005v8.0
		Sun Java Platform Standard Edition JDK 1.5 or 1.6	0WINDOWS1500
Windows 2003 x64 Edition	x64	Visual Studio 2005 SP 1	x64WINDOWS2005
		Visual Studio 2008 SP 1 (C++ /CLI, C# 2.0 or 3.0)	x64WINDOWS2005v8.0
		Visual Studio 2008 SP 1	x64WINDOWS2005
		Sun Java Platform Standard Edition JDK 1.5 or 1.6	x64Windows
Windows Server 2008 R2 x64 Edition	x64	Visual Studio 2008 SP 1 (C++ /CLI, C# 2.0 or 3.0)	x64WINDOWS2005v8.0
		Visual Studio 2010	x64WINDOWS1010
		Visual Studio 2008 (C++ /CLI, C# 2.0 or 3.0)	x64WINDOWS2005v8.0
		Sun Java Platform Standard Edition JDK 1.5 or 1.6	x64Windows
Windows Vista	x86	Visual Studio 2005 SP 1	0WINDOWS2005
		Visual Studio 2008 SP 1 (C++ /CLI, C# 2.0 or 3.0)	0WINDOWS2005v8.0
		Visual Studio 2008 SP 1	0WINDOWS2005
		Visual Studio 2008 SP 1 (C++ /CLI, C# 2.0 or 3.0)	0WINDOWS2005v8.0
		Sun Java Platform Standard Edition JDK 1.5 or 1.6	0WINDOWS1500

Table 1.11 **Windows Platforms**

Operating System	CPU	Compiler or Software Development Kit [1, 2]	RTI Architecture
Windows Vista 64-bit Edition	x64	Visual Studio 2005 SP1	x64Win64VS2005
		Visual Studio 2005 SP1 (C++/CLI, C#/F# or 9.0)	x64Win64dotNetCLI2
		Visual Studio 2008 SP1	x64Win64VS2008
		Visual Studio 2008 SP1 (C++/CLI, C#/F# or 9.0)	x64Win32dotNetCLI2
		SuperLow-Platform: Standard Edition (DK 1.5 or 1.6)	x64Win64pfs
Windows XP Professional [3]	pfx	Visual Studio 2005 SP1	pfxWin32VS2005
		Visual Studio 2005 SP1 (C++/CLI, C#/F# or 9.0)	pfxWin32dotNetCLI2
		Visual Studio 2008 SP1	pfxWin32VS2008
		Visual Studio 2008 SP1 (C++/CLI, C#/F# or 9.0)	pfxWin32dotNetCLI2
		SuperLow-Platform: Standard Edition (DK 1.5 or 1.6)	pfxWin32pfs
Windows XP Professional x64 Edition	x64	Visual Studio 2005 SP1	x64Win64VS2005
		Visual Studio 2005 SP1 (C++/CLI, C#/F# or 9.0)	x64Win64dotNetCLI2
		Visual Studio 2008 SP1	x64Win64VS2008
		Visual Studio 2008 SP1 (C++/CLI, C#/F# or 9.0)	x64Win32dotNetCLI2
		SuperLow-Platform: Standard Edition (DK 1.5 or 1.6)	x64Win64pfs

1. On Windows XP, if you are using JDK 5.0 and want to use Intel's Hyper-Threading technology, use JDK 5.0 Update 6 or later. Prior to JDK 5.0 Update 6, the JVM ignored Hyper-Threading. (If you meet our Optimal Install 1.5.0_05, you should disable Hyper-Threading.)
2. The RTI .NET assemblies are supported for both the C++/CLI and C# languages. The type-support code generated by rtiddsgen -language C++/CLI is compiling for generated type-support code requires Microsoft Visual C++. Calling the generated code from C# is supported.
3. Windows XP does not support IP_TOS setsockopts changes, see made. See http://support.microsoft.com/kb/248611, http://www.winsocketd.com/Archive/Unbounding/Issues/2007-03/2_0.html#Jon_Mitchell-qos.

2 Compatibility

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

- Wire Protocol Compatibility (Section 2.1)
- Code Compatibility (Section 2.2)
- Data-Format Compatibility and Extensibility (Section 2.3)
- ODBC Database Compatibility (Section 2.4)
- What's Fixed in 4.5e (Section 2.5)

2.1 Wire Protocol Compatibility

2.1.1 General Information on RTPS (All Releases)

RTI Data Distribution Service communicates over the wire using a 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 standardized and submitted to 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 RTI Data Distribution Service 4.5

RTI Data Distribution Service 4.5x is compatible with 4.2 – 4.4, except as noted below.

2.1.2.1 RTPS Versions

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

2.1.2.2 double, long long, unsigned long long or long double Wire Compatibility

If your *RTI Data Distribution Service* 4.4x or higher application's data type uses a 'double', 'long long,' "unsigned long long," or "long double," it will not interoperate with *RTI Data*

Distribution Service applications built with version 4.2e or lower, unless you use the **-unsafe 60,payload** flag when generating code with *rtidlgen*.

2.1.2.3 Sending 'Large Data' between 4.6d and Older Releases

The 'large data' format in *RTI Data Distribution Service* 4.2e, 4.3, 4.6b and 4.6c 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 has been resolved, starting with *RTI Data Distribution Service* 4.6d. As a result, by default, large data in *RTI Data Distribution Service* 4.6d and higher is not compatible with older versions. You can achieve backward compatibility by setting the following properties to '1'.

```
dds.data_writer.protocol.use_2_1_large_data_format
dds.data_reader.protocol.use_2_1_large_data_format
```

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

For example:

```
<qos_profile name="
  <qos_snippet>
    <participant_qos>
      <property>
        <value>
          <element>
            <name>
                dds.data_writer.protocol.use_2_1_large_data_format
            </name>
            <value>1</value>
          </element>
          <element>
            <name>
                dds.data_reader.protocol.use_2_1_large_data_format
            </name>
            <value>1</value>
          </element>
        </value>
      </property>
    </participant_qos>
  </qos_snippet>
```

2.2 Code Compatibility

2.2.1 General Information (All Releases)

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

RTI Data Distribution Service 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 need to send and receive messages. To create code for a data type, *RTI Data Distribution Service* 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 dave 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 *rtiddsgen* on the same IDL file you used to generate the IDL. This process will regenerate the header and source files, which can then be compiled along with the rest of your application.

2.2 Release-Specific Information for RTI Data Distribution Service 4.5

2.2.2.4 Type Support and Generated Code Compatibility

- **long long Native Data Type Support**

  Starting with release 4.5c, we assume all platforms natively support the 'long long' data type. There is no longer a need to define RTI_COR_SESSION_LONG_LONG in for 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 *RTI Data Distribution Service* 4.5 or higher application's data type uses a 'double', 'long long' or 'unsigned long long' data type, it will not be backwards-compatible with *RTI Data Distribution Service* applications built with version 4.2e or lower, unless you use the **-unsafeAlignment** flag when generating code with *rtiddsgen*.

- **Changes in Generated Type Support Code**

  The *rtiddsgen*-generated type-support code for user-defined data types 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

In *RTI Data Distribution Service* 4.3, keys were serialized with the incorrect byte endianness when using Java or .NET. This bug affected the cross-language interoperability in functions 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 version 4.3e-rev. 01 and starting in .NET in 4.4b.

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

This issue does not impact systems using 4.4 and higher versions, or 4.2 and lower versions of *RTI Data Distribution Service*.

❑ Data Type with Similar 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 field keyed by a long as the key):

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

Specifically, all samples will be received in these 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 4.4d or higher, this can happen only 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 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 4.6x, 4.0x or 4.1x application, you can resolve the backwards-compatibility problems by regenerating the code with version of rtidlgen distributed with *RTI Data Distribution Service* 4.4d. You can also upgrade your whole system to this release.

22.2.5 Other API and Behavior Changes

- **New set_instance_replacedf() method on DataWriterListener**

  Starting with release 4.5c, 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 release 4.5c, all the 'count' data types in DataReaderCacheStatus, DataReaderProtocolStatus, DataWriterCacheStatus and DataWriterProtocolStatus have been changed from 'long' to'long long' in the C, C++ and .NET APIs in order to report the correct value for DDS applications that run for very long periods of time. If you are using these types with a previous release that is sensitive to these fields, data type changes may be necessary.

- **Changes in RtpsReliableWriterProtocol_t**

  Starting in release 4.4c, two fields in DDS_RtpsReliableWriterProtocol_t have been renamed:

  - Old name: *disable_positive_acks_decrease_sample_keep_duration_scalar*
    New name:*disable_positive_acks_decrease_sample_keep_duration_factor*
  - Old name: *disable_positive_acks_increase_sample_keep_duration_scalar*
    New name: *disable_positive_acks_increase_sample_keep_duration_factor*

  In releases prior to 4.4c, the NACK-only feature is not supported on platforms without floating-point support. Other 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. Starting with release 4.4c, the NACK-only feature has been reimplemented to use fixed-point arithmetic and the new DDS_Long 'factor' fields noted above, which replace the DDS_Double "scalar" fields.

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

  Starting with release 4.4b, by default, *RTI Data Distribution Service* is less restrictive (compared to older releases) on the reader 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, **reject** will ensue.

  If you are upgrading from *RTI Data Distribution Service* 4.4a or lower, and the application you are upgrading relied on the old-*like* or *strict* timestamps that 'reset 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 set the new **source_timestamp_tolerance** to 0. Alternatively, if the designer of your system had a particular reason for using BY_RECEPTION_TIMESTAMP (perhaps because you did not want to match reader with BY_SOURCE_TIMESTAMP) if you need to keep the BY_RECEPTION_TIMESTAMP setting, there is no QoS-setting that will give you the exact same behavior on the writer side as the previous release.

  Starting with release 4.4b, by default, *RTI Data Distribution Service* is more restrictive (compared to older releases) on the reader side with regards to source and reception timestamps of a sample of 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_PROFILE.xml)**

  Starting with release 4.4d, the default XML QoS Profiles file has been renamed and is installed in a new directory:

- OLE location/name:
  $NDDSHOME/resource/cmd/NDDS_QOS_PROFILES. and
- Now location/name:
  $NDDSHOME/resource/qos_profiles_<version>/cmd/
  NDDS_QOS_PROFILES.example.cmd (where <version> can be 4.6d, 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.cmd** to **NDDS_QOS_PROFILES.cmd** (i.e., by default, even if your **NDDSHOME** environment variable is set, this QoS profile is not used.) See Section 15.2 in the *User's Manual* for details.

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

  Starting with release 4.6d, 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 Only**

  Starting with release 4.5c, the constructor for SampleInfoSeq has been changed from DDS.SampleInfoSeq(int maximum) to DDS.SampleInfoSeq(int2 maximum). This was to make it consistent with other sequences.

- **Default Send Window Size Changed to Infinite**
  - Starting with release 4.5d, the send window size of a *DataWriter* is set to infinite by default. Prior to release 4.5d, the default values of two fields in *DDS_ReliabilityRestrictedProtocol* (**max_send_window_size**, **max_send_window_size**) to be DDS_LENGTH_UNLIMITED.
  - In release 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 pre-established non-blocking default behavior, the send window size has been changed to be infinite by default starting in release 4.5d.
  - Users wanting the performance benefits of a finite send window will now have to configure the send window explicitly.

2.3 Data-Format Compatibility and Extensibility

With *RTI Data Distribution Service*, you can define your own data types, which will be used to communicate between applications on a network.

Developers sometimes extend or modify their data types. When new fields are added, it is a common requirement that applications that use the extended types must still communicate with applications using the old types.

To accomplish this, RTI provides a dynamic data API, which allows applications to define data types at runtime without code generation. Specifically, this API supports the concept of a *sparse* type, one for which every data sample need not contain a value for every field defined in the type. By dynamically defining a type as sparse, an application can add fields to the type without breaking existing components that may be unable to fill in these new fields.

For customers that may want to handle extensibility in a more custom way, such as with XML payloads or custom serialization and deserialization, RTI provides built-in string and opaque data types.

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 features 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 TimeTen database has been removed.

To use MySQL, you will also need the MySQL ODBC 5.1x (or higher) driver. For non-Windows platforms, UnixODBC 2.2.12 (or higher) is also required.

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

- **AIX:** pSeries5.3a0-0.0, x86Aix5.3a0-0.0
- **Linux:** i86Linux2.6gcc3.4.3, x64Linux2.6gcc3.4.3, i86Linux2.6gcc3.1.1, x64Linux2.6gcc3.1.1, i86Linux2.6gcc4.1.3, x64Linux2.6gcc4.1.3
- **Solaris:** sparcSolr2.11hgcc3.4.2 and sparcSolr2.11hgcc3.4.2

- **Windows:** *libWinx32VS2005*, *libWinx32VS2008*, *libWinx32VS2010*, and *winSHmem8.VS2010*

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

3 What's Fixed in 4.5e

This section includes:

3.1 Fixes Related to Usability

3.1.1

Improved Logging

The logging facility has been improved in the following ways:

- Issues with inconsistent QoS settings will now be logged as errors instead of warnings and will include the name of the QoS policy in which the error was detected. [RTI RTI-# 1312]
- Transport properties are now validated; a warning will be logged if an invalid transport property is specified. [RTI RTI-# 439]

- The error message logged for a compilation error related to a content-filter expression for a ContentFilterProfile will now specify the actual error detected in the filter expression. [RTI RFI # 296, RTI Bug # 12622]
- The locations of all XML files used for loading QoS Profiles are now logged as warnings. [RTI RFI # 345]

3.1.2 Serialization of Sequences Exceeding Maximum Size not Reported on Error

If a DataWriter wrote a sample of a type containing a sequence field and that sequence's length was larger than the maximum specified in the type definition, the middleware may not have detected the error. This problem has been resolved; the error condition is now detected and the write operation will fail.

[RTI Bug # 14874 and 14852]

3.1.3 XML Schema Supplied for QoS Profiles not Conformant to W3C Specification

The XML schema provided to describe QoS profiles (in **$NDDSHOME/resource/qos_profiles_5.0/dds/benchmark_qos_profiles.xsd**) was not valid. The NSD validation process would will fail on error:

```
Value '[a-zA-Z0-9\^_]+' is not a valid XML regular expression.
```

This problem has been resolved.

[RTI Bug # 14840]

3.2 Fixes Related to the .Net API

3.2.1 Unhandled Exceptions from .Net Type Plugin

Exceptions thrown by a .Net type plugin were not caught within the type-plugin or the middleware. This resulted in unhandled exceptions that would ungracefully terminate a user application. The issue has been resolved by having the middleware catch (and print) the exceptions.

[RTI Bug # 13867]

3.2.2 System.AccessViolationException from CH on 64-bit Windows

When running a CH RTI Data Distribution Service application on a 64-bit Windows system (Windows Vista, Server 2003 or 2008) using .Net framework version 2.0, the following exception may have been encountered:

```
System.AccessViolationException: Attempted to read or write prot
ected memory. This is often an indication that other memory is cor
rupt.
```

This could happen when the application consumed a large amount of memory and was manipulating 1 DDS_InstanceHandle_t structure. This problem has been resolved.

[RTI Bug #14023]

3.2.3 FlowController Names Incorrectly Defined in .net API

Setting *DataWriterQos.publish_mode.flow_controller_name* to either FIXED_RATE_FLOW_CONTROLLER_NAME or ON_DEMAND_FLOW_CONTROLLER_NAME had no effect and the default FlowController (DEFAULT_FLOW_CONTROLLER_NAME) was used instead. This issue has been resolved.

[RTI Bug #13356]

3.2.4 Buffer Creation Exception when Using Listeners in .Net API

Applications installing listeners on the *DataReader* or the *DataWriter* may have experienced a buffer overrun exception. This problem has been resolved.

[RTI Bug #13636]

3.3 Fixes Related to Communication over Loopback

3.3.1 Loopback Address was Limited to 127.0.0.1

A valid loopback address is in the range 127.0.0.1 to 127.255.255.255. However, *RTI Data Distribution Service* previously only recognized 127.0.0.1 as a loopback address. This limitation has been removed.

[RTI Bug #13865]

3.3.2 Participants on Same Node did not Communicate if One Used Loopback Instead of Shared Memory

DDS applications running on the same node failed to communicate when one of the *DomainParticipants* disabled shared memory but used UDP loopback. In this scenario, communication between the applications through UDP loopback would be the expected behavior. However the second *DomainParticipant* (using both shared memory and UDP loopback) would automatically disable UDP loopback as an optimization to avoid sending duplicated data. You can avoid this situation by setting the UDP transport property **ignore_loopback_interface** to 0.

Now when this situation is detected, DDS will print the following warning message:

```
PRESParticipant_assocAndAddRemotePeriodicParticipant(checking) remote participant
on same host (with name "Hello World") has disabled shared memory
```

```
and this participant ignores UDP loopback. To allow communication,
set ignore_loopback_interface to 0
```

[RTI:Bug #14102]

3.3.3 UDPv4 Multicast Loopback Crosstalk—Linux Systems Only

If you have two participants on the same Linux host and in the same domain, both listening for discovery traffic over multicast, they would discover each other, regardless of the multicast address to which they are listening. This is because on Linux systems, the implementation for multicast loopback ignores the group address, so all multicast traffic is received, rather than the correct behavior of receiving messages only for the address the current process is subscribed to.

To prevent spurious multicast crosstalk, the built-in UDPv4 transport implementation has been updated to uniquely configure a port and create a receive thread for each multicast address used by the DomainParticipant. This behavior is controlled by property in NDDS_Transport_UDPv4_Property_t, named **enum_multicast_receive_preferred**, where not meaning the receive resource (the default) prevents multicast crosstalk.

[RTI:Bug #11054]

3.4 Fixes Related to Reliability Protocol

3.4.1 Slow Performance when Resending Data to Reliable DataReaders

Based on its configuration, a late-joining reliable DataReader may be entitled to receive all samples that have been published in the past. The DataWriter responding to its request may easily may have taken longer to complete than expected, a bug with the DataWriter caused an incorrect interpretation of the DataReader's NACK message, consequently the DataWriter did not send all the samples requested, with potentially all requested samples not being sent. The DataReader would have to wait for the next HEARTBEAT to request the unsent samples; increasing the rate of periodic and piggyback HEARTBEATS may have shortened the unintended delay in receiving the samples, but it would not have solved the root cause. The problem has been resolved.

[RTI:Bug #13630 and #13962]

3.4.2 Reliable Non-RTI DataWriter Possibly Mistaken for Best-Effort DataWriter

Previously during simple endpoint discovery, when interoperating with a non-RTI, DDS-compliant implementation that does not send Reliability information with its Publication built-in topic data, RTI Data Distribution Service incorrectly interpreted the Reliability kind of the corresponding DataWriter to be Best-Effort. The correct interpretation should be the default DataWriter Reliability kind, which is Reliable. Consequently, rel-

able communication may not have been established due to the erroneous assumption of inconsistent QoS. This problem has been resolved.

[RTI Bug # 14089]

3.4.3 Possible Segmentation Fault if max_send_window_size Limit Reached

When using non-GIL data types (such as CORBA or Dynamic Data), you may have seen a segmentation fault if a call to **write()** required blocking because the **max_send_window_size** resource limit is reached. This problem has been resolved.

[RTI Bug # 13830]

3.4.4 Periodic Heartbeats not Sent Continuously to Destinations with Disabled Positive ACKs

If **disable_positive_acks** was set in either the DataWriterProtocolQosPolicy or the DataReaderProtocolQosPolicy, the DataWriter involved did not continuously send periodic HEARTBEAT messages to its DataReaders. Instead, it only sent an initial "periodic" HEARTBEAT, it did not sent any more periodic HEARTBEATs. This problem has been resolved.

[RTI Bug # 14112]

3.4.5 High CPU Consumption when Positive ACKs Disabled

If **disable_positive_acks** was set in either the DataWriterProtocolQosPolicy or the DataReaderProtocolQosPolicy, the DataWriter involved may have experienced high CPU consumption. This problem has been resolved.

[RTI Bug # 13990]

3.5 Other Fixes Related to RTPS Protocol

3.5.1 Data Unnecessarily Dropped if Not 4-Byte Aligned

RTI Data Distribution Service previously assumed that received data was aligned on a 4-byte boundary. This implementation was stricter than necessary to satisfy RTPS wire interoperability specification; in fact, the data of the last submessage within an RTPS message is not required to be 4-byte aligned. As a result, such data was dropped by *RTI Data Distribution Service* when according to RTPS specifications it should have been accepted. This issue has been resolved.

[RTI Bug # 13891]

3.5.2 Data Fragments Not Always 4-Byte Aligned

The RTPS wire protocol specifies that submessages must be 4-byte aligned. This was not always done in the implementation of data fragments of large samples. Consequently,

some data fragment rudimentary were not 4-byte aligned and these data fragments would not have been received correctly by other interoperable DDS implementations. This alignment issue has been corrected.

[RTI-Bug 4136]

3.5.3 Unnecessary Traffic for Dispose Samples when serialize_key_with_dispose is TRUE

Previous releases sent unnecessary traffic on the network for dispose samples if **serialize_key_with_dispose** in the DataWriterProtocolQosPolicy was set to TRUE. This problem has been resolved.

[RTI-Bug 4140.2]

3.5.4 DataReader Crashed When Interoperating with Non-RTI DataWriter

Previously a DataReader interoperating with a non-RTI, DDS-compliant middleware DataWriter could crash. This was due to an out-of-bounds memory read, triggered when the DataReader interpreted a non-standard sublength (i.e. a vendor-specific extension). The problem has been resolved.

[RTI-Bug 4140.25]

3.5.5 Possible Memory Corruption when Sending Large Data in BEST_EFFORT Mode

Memory may have been corrupted if the number of DataFrames matching a best-effort DataWriter was greater than **participant_qos.resource_limits.max_gather_destinations**. This problem has been resolved.

[RTI-Bug 4136?]

3.6 Fixes Related to QueryConditions and ReadConditions

3.6.1 Multiple Instances Returned by Read/Take APIs with Query Conditions

When using one of the read/take APIs with QueryConditions, the middleware improperly returned more than one instance's data (if spare was available in the requested length). This applied to **read_w_condition()**, **take_w_condition()**, **read_instance_w_condition()**, and **take_instance_w_condition()**, **read_next_instance_w_condition()**, and **take_next_instance_w_condition()**. This problem has been resolved.

[RTI-Bug 4140.30]

3.6.2 QueryCondition Failed to Return All Matching Samples of a Keyed DynamicData Type

When using QueryConditions based on keyed types created using DynamicData, where the query expression is based solely on key members, or where the query expression is

promiscuous (i.e., "*", meaning match all samples), the QueryCondition did not recognize all matching samples, and thus did not return all matching samples.

This problem has been resolved. Now, such QueryConditions (those based on layered types created using DynamicData, where the query expression is based solely on key members, or where the query expression is promiscuous) will recognize and return all matching samples.

[RTI Bug # 14132]

3.6.3 Positive Memory Corruption and Incorrect Behavior for QueryConditions

When using QueryConditions, if resource limits (such as **max_samples** in the ResourceLimitsQosPolicy) were set to non-default values, the may have resulted in corrupted QueryCondition sample queues and the return of non-matching samples, as well as possible memory corruption.

[RTI Bug # 14104]

3.6.4 Incorrect Behavior from read_w_condition() if max_samples not DDS_LENGTH_UNLIMITED

In the previous release, if **read_w_condition()** was called with a **max_samples** parameter set to a finite value (not DDS_LENGTH_UNLIMITED), there was an issue when samples from different instances were interleaved in the reader queue (e.g., samples from each instance were not contiguous), and they were read with a SAMPLE_STATE_NOT_READ mask.

The internal optimization did not advance the FIRST_NOT_READ property in the case of a partial read (not all samples read), with interleaved samples. Subsequently, **read_w_condition()** erroneously indicated that no more NOT_READ samples were contained in the queue.

This problem has been resolved by correcting the maintenance of the internal FIRST_NOT_READ indicator, so that reading with SAMPLE_STATE_NOT_READ now correctly returns the proper samples, for all known cases of interleaved samples, out-of-order reading partial reads, etc.

[RTI Bug # 13963]

3.7 Fixes Related to rtiddsgen

3.7.1 Compilation Warning on Windows-64-Bit Platforms

In previous releases, compilation of the generated code on Windows-64-bit platforms may have resulted in this warning:

```
warning C4267: conversion from size_t to unsigned int possible loss
of data.
```

This problem has been resolved.

[BT:Bug # 13765]

3.7.2 IDL with Typedefs of Arrays or Sequences of Enums Caused Compiler Errors—Net Only

IDL that contained a typedef of an array or sequence of enums resulted in compilation errors when using .Net. For example, the following IDL resulted in an error:

```
module IDL
{
    enum SampleEnum
    {
        VALUE_ONE,
        VALUE_TWO,
        VALUE_THREE
    };
    typedef sequence<SampleEnum, 10> SampleEnumSequence;
    typedef SampleEnum SampleEnumBoundArray[10];

    struct SampleEnumStruct
    {
        SampleEnumSequence valueSequence;
        SampleEnumBoundArray valueBoundArray;
    };
};
```

This problem has been resolved.

[BT:Bug # 13846]

3.7.3 XML Type-Definition Files Containing Boolean Literals Parsed Incorrectly by rtidlgen

rtidlgen did not correctly parse XML type-definition files that contained boolean literals. For example, the following XML resulted in generated code that would not compile in C:

```
<?xml version="1.0" encoding="UTF-8"?>
<types>
    <module name="MyType">
        <dlrservicemsg xt:type="boolean"/>
        <struct>
            <boolliteralmsg xt:value="true"/>
            <member name="m1" type="long"/>
        </struct>
        <struct>
            <trueBoolliteralmsg xt:value="false"/>
            <member name="m1" type="long"/>
```

```
        </owner>
    </member>
</typeref>
```

This problem has been resolved.

[RTI Bug # 13915]

3.7.4 Compile Error Caused by Empty Defined Value Type in IDL

If a field from a base valuetype was used in a ContentFilteredTopic for an empty derived class, the following error was generated:

```
structure has no member\' \'content filter compile error 1
```

For example, the error occurred if you had the following IDL:

```
valuetype B {
    public long field;
};
valuetype D : B {
};
```

and then tried to create a ContentFilteredTopic as follows:

```
DDSContentFilteredTopic *cft = NULL;
rfs = participant->create_contentfilteredtopic(
    "ContentFilteredTopic", topic, "field1 >= %0", parameters);
```

where the topic is of type 'D'. There were no problems if D had a field.

This problem has been resolved so that it is now possible to create SQL filters using types that contain no concrete members at the top level (i.e., contain only valuetypes and/or opaque types at the top level). Types that consist solely of valuetypes at the top level can be filtered as long as a concrete type exists somewhere in the parent type (either in the type itself, or in the base types of valuetypes enumerated in the parent).

[RTI Bug # 14036]

3.8 Fixes Related to TCP Transport

3.8.1 Failure to Re-establish Communication if Participants using TCP were Destroyed and Recreated

Two DDS applications using the TCP transport would stop communicating if the participant in one of the applications was deleted and created again while the participant in the other application kept running. This problem has been resolved.

[RTI Bug # 13881]

3.8.2 Calls to send() Blocked when Available Bandwidth Exceeded

Applications sending large amounts of data over TCP may have been blocked indefinitely if they exceeded the available bandwidth.

To prevent **send()** from blocking in this manner, there is a new TCP transport property, **send_max_wait_sec**, that limits the time a thread can be blocked on a **send()** operation. The default value is 3 seconds.

[RT:Bug #12957]

3.8.3 High Latency of TCP Transport—Linux Systems Only

Applications may have experienced high latency when using the TCP transport over large distances. This problem has been resolved.

[RT:Bug #14855]

3.9 Fixes Related to Status

3.9.1 Segmentation Fault when Retrieving Cache Status for Disabled DataReader or DataWriter

Attempting to retrieve the cache status from a disabled *DataWriter* or *DataReader* before enabling them may have resulted in a segmentation fault. This is a valid operation and has been fixed.

[RT:Bug #12957]

3.9.2 DataReader's Cache Status Not Updated After Take

The *DataReaderCacheStatus*' **sample_count** (which represents the number of samples available in a DataReader's queue) was not being updated after samples were taken from the queue. Therefore the retrieved **sample_count** may have been larger than the actual number of queued samples. This problem has been resolved.

[RT:Bug #13052]

3.9.3 DataReader's Protocol Status Not Updated Correctly

The *DataReaderProtocolStatus*' **filtered_sample_count** (which represents the number of user samples prematurely filtered by a local *DataWriter* due to *ContentFilteredTopic*s) was not being updated correctly—for keyed or unkeyed data. Users may have seen the count increment once and then be stuck at 1. This problem has been resolved.

[RT:Bug # 13634]

3.10 Fixes Related to ContentFilteredTopics

3.10.1 Filter-Related Statuses at DataReader Not Correctly Updated

The fields in DataReaderFilterStatus related to filtered samples (e.g., **filtered_sample_count**) were not correctly updated in all cases for content-based filtering or time-based filtering. The retrieved values of these fields were likely less than the actual number of samples (or bytes) filtered by a *DataReader*. This problem has been resolved.

[RTI Bug # 12427]

3.10.2 Fixed an Segmentation Fault when Changing Content Filter Parameter

In the previous release, you may have seen a segmentation fault after calling **set_expression_parameters()** or **append_to_expression_parameters()** on a ContentFilteredTopic if the DataReader was filtering samples when the call was made. This problem has been resolved.

[RTI Bug # 13819]

3.10.3 Content-Filtered Topic Creation Failed when Code Generated with rtiddsgen -package -python -java API Only

The creation of a ContentFilteredTopic in Java failed if the type code was generated with the **-package** option in rtiddsgen.

In this scenario, you would have seen an error such as the following:

```
[0x2241]RADAR ETPacket::Subscriber.Failing
[0x2241]RADAR ERtipart_presentContentFilteredTopic:create_filter
failed.
[0x2241]RADAR ETPacket::Subscriber.Failing
[0x2241]RADAR ETPacket::Subscriber.Failing
[0x2241]RADAR ERtiddsmodelPlugin_create2::create DDS_ContentFilteredTopic
failed.
```

This problem has been resolved.

[RTI Bug # 13866]

3.11 Fixes Related to Batching

3.11.1 Unexpected Dynamic Memory Allocation When DataWriter Used Batching

Previously, a *DataWriter* using batching (**writer_qos.batch.enable** = DDS_BOOLEAN_TRUE) would dynamically allocate a few bytes of memory for each batch written. This problem has been resolved. Now the *DataWriter* will preallocate the relevant memory.

[RTI Bug # 13660]

3.11.2 Asynchronous Batch Flushing not Triggered in Some Cases

Asynchronous batch flushing (configured with **max_flush_delay** in the BatchQueuePolicy) was not rescheduled in some situations if the *DataWriter*'s queue was full at the moment of flushing. This could have resulted in samples not being sent unless the *DataWriter* received a new sample or explicitly called **flush()**.

[RTI Bug #14131]

3.12 Other Fixes

3.12.1 Memory Leak When Calling DomainParticipant's set_listener() – 5.0c Rev. 01 Only

When using version 4.5c Rev. 01, calls to the *DomainParticipant*'s **set_listener()** function caused a memory leak. This problem has been resolved.

[RTI Bug #13640]

3.12.2 Error Reported on Platforms where Hostname Address Not Obtainable

Depending on your WinsProtocol QoS policy settings, *RTI Data Distribution Service* may need to get the MAC address of your machine. However, in previous releases, it always reported an error if it was unable to do so. This error could be misleading in some situations. On some platforms retrieving the MAC address is not supported on require special privileges. On those platforms, you may have seen the following error message, even if your QoS settings did not require getting the MAC address:

```
DE RTIOsapiBsd_getMSAddrbyIf() failure, error GetAdaptersAddresses
```

The message is benign, it is just notification that the MAC address could not be obtained.

In this release, *RTI Data Distribution Service* will no longer attempt to get the MAC address unless it is needed.

[RTI Bug #13869]

3.12.3 Crash in Custom Discovery Plugin When Using Partitions

When creating a custom Discovery plugin, if DDS partitions are set in either the *DataWriter* or *DataReader* being discovered, the application may have crashed. This problem has been resolved.

[RTI Bug #13800]

3.12.4 Locators not Propagated when Using Disabled DataReader and Custom participant_id

If a *DataReader* was created in a disabled state and its *DomainParticipant* had an explicitly configured **participant_id**, then neither its unicast nor multicast locators were propa-

gated during the discovery process. This effectively caused the *DataReader* to only receive over its *DomainParticipant*'s default unicast locator. This problem has been resolved.

[RTI Jira #13673]

3.12.5 Incorrect Writer-Side Time-Based Filtering

Writer-side time-based filtering was incorrectly applied to samples written backwards in time. When using the default settings for the TimeBasedFilter QoS policy and writer-side time-based filtering, a *DataReader* would not receive any samples that were written with decreasing source timestamps. This problem has been resolved.

[RTI Jira #12995]

3.12.6 Unexpected Behavior when Using Defaults for Mask and Priority of Threads Created DDS—Linux Systems Only

By default, the priority and scheduling policy (set with the mask attribute) of the threads created by *RTI Data Distribution Service* on Linux systems should be set to the default values assigned by the POSIX call, **pthread_attr_init()**.

However, due to a bug in GLIBC, the priority and scheduling policy of the middleware threads are inherited from the parent process, which leads to different behaviors depending on the settings of the parent process.

This issue has been resolved by explicitly setting the priority and scheduling policy to the values returned by the call to **pthread_attr_init()**.

[RTI Jira #14120]

4 Known Issues

4.1 Multicast over UDPv6 only Supported with Single Link-Local IPv6 Address

Multicast over the UDPv6 transport is supported only when the network interface have the link-local IPv6 address and no other IPv6 addresses.

In a configuration where a network interface has more than one IPv6 address (e.g., an address assigned by SLAAC in addition to the link-local address), UDPv6 multicast is not supported at this time.

4.2 Estimate of Message Overhead For Large Data May Be Exceeded

RTI Data Distribution Service underestimates the message overhead of samples with large data payloads (on the order of 64 KB). If not properly compensated by the transport's **message_size_max** property, the send operation will fail.

To work around this issue, **message_size_max** should be sized to account for this overhead. As a rule-of-thumb, 64 KB requires an additional 1024 bytes of overhead.

[RTI Bug #11990]

4.3 Writer-side Filtering May Cause a Deadline to be Missed

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

[RTI Bug #11745]

4.4 Disabled Interfaces on Windows Systems

The creation of a *DomainParticipant* will fail if no interface is enabled *and* the **Discovery-QosPolicy.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 *DomainParticipant* will fail if the TransportMulticastQosPolicy contains a UDPv4 or UDPv6 multicast address.

4.5 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 Bug #11342]

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 or C, ASPn, or CORBA-compatible C++ type plugins.

[RTI Bug #12686]

4.7 Multi-dimensional Arrays of Sequences Not Supported when Generating .Net Code

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

For example:

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

[RTIBug #13086]

4.8 License file may not be found if NODISCME Set with Qualation Marks

Some distributions of *RTI Data Distribution Service* require a license. One of the places where *RTI Data Distribution Service* may look for a license is the file **rti_license.dat** in the directory specified by the environment variable NDDSHOME. However, if you have set NDDSHOME to a directory containing a space, such as **"C:\Program Files\RTI\ndds.4.5e"**, the license file will not be found there. Instead, you should set NDDSHOME without using qualation marks (such as **C:\Program Files\RTI\ndds.4.5e**).

For more information on where *RTI Data Distribution Service* may look for a license file, see the *RTI Data Distribution Service Getting Started Guide*.

[RTIBug #13677]

4.9 Errors when Using Monitoring Library and Long Names/Values in PropertyQosPolicy

If your DDS application links with *RTI Monitoring Library* and the PropertyQosPolicy of a *DataReader*, *DataWriter*, or *DomainParticipant* has a name longer than 127 characters or a value longer than 511 characters, you will see an error in the logs relating to your DDS application.

```
PRSSPublisherRoutersPrivate_initializeSample:initialize
NetworkStatusMemoryPlugin_addNetworkPlatform:initialize sample
NetworkStatusMemoryPlugin_getEntry:add virtual sample to semaphore
NetworkStatusMemoryPlugin_getEntry:exit
PRSSPublisherRoutersPrivate_addNetworkPlatform:add_sample
PRSSPublisher_write:initialize addNote
RTIDefaultRoutersPersistenceWriter_sampleSendPublishInstance:write
RTIDefaultRoutersPublisher_onSendWrite:fini/publish writer done
```

As a work-around, you can set the following monitoring configuration properties to 'true' to prevent the PropertyQueFolicy being sent as part of the monitoring data for certain type of entities:

```
ui.monitor.config.ship_routing_properties
ui.monitor.config.ship_reader_properties
ui.monitor.config.ship_writer/ipset_properties
```

[RT:Bug # 13625]

4.10 Changes Required when Using More than 5 Content-filter Parameters

The **contentfilter_property_max_length** (in the DomainFilterSpecificationLimitSpecifolicy) needs to account for the number of characters that each part of the ContentFilterSpec data (SrcPattern, CPname, CPfuge, pattern, etc) plus one null-termination character for each string.

However, if more than five content-filter parameters are specified, the calculation above may not be sufficient. In this case, when specifying the **contentfilter_property_max_length**, you need to account for at least seven more characters for each extra parameter (over the 5th one).

Otherwise, you may see this error when creating the DataReader:

```
DCPSInfoRepo,ae,ae1:ae2:ae3:ae4:ae5:ae6:ae7:ae8:ae9:ae10:
ae11:ae12:
PRESEntitiesHistoryCore::publishLifeSample:localize
PRESEntitiesHistoryCore::publishLifeSample:publishLifeSample
BrokerEntityMemoryPlugin::getEntity:add virtual sample to sensitive
BrokerEntityMemoryPlugin::addSample:lpc entry
PRESEntitiesHistoryCore::addLive:add_sample
PRESEntitiesHistoryCore::addLive:addEntity
BrokerEntityMemoryPlugin::processPlugin:PRESEntitiesHistoryCore::addLive:
add failed
```

[RT:Bug # 13741]

4.11 DataReaders Using GROUP_PRESENTATION access_scope May Lose Samples if DataWriters Lose Backlog

DataReaders whose PresentationQosPolicy's **access_scope** is set to DDS_GROUP_PRESENTATION_QOS may lose samples if their matching DataWriters are using batching.

This issue only occurs if the DataWriters are configured to use virtual HeartBeats (VHBs) (periodic virtual HBs or piggyback virtual HBs), which are configured with these two parameters:

- writer_qso.protocol.rtps_reliable_writer.virtual_heartbeat_period
- writer_qso.protocol.rtps_reliable_writer.samples_per_virtual_heartbeat

[RTI Bug # 14177]

4.12 Issues with Dynamic Data

- The conversion of data by member-access primitives (**get_X0** 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 in smaller types on a subset of platforms. Conversion to 64-bit long longs from a 32-bit or smaller type is supported on platforms that have 64-bit long long data representation, and any additional 64-bit structures. Conversion to 128-bit long doubles from a float or double is only supported on Solaris SPARC architecture.

  [RTI Bug # 12647]
- Dynamic Data cannot handle a union with a discriminator that is set to a value which is not defined in the type.

  [RTI Bug # 12853]
- Dynamic Data may have problems restoring variable-size members that are >= 64k in size. In this case, the method **last_X0** or **unlink_complex_member()** will fail with the error "queryd: stored member records 65535 bytes." Note that it is not possible for a member of a sparse type to be >= 64k.

  [RTI Bug # 12867]
- Topics of DynamicData types that contain bit fields are not supported by rtiddsspy.

  [RTI Bug # 13948]
- Dynamic Data does not support set-or-value assignment of members that are longer than 65,535 bytes. In this situation, the DynamicData API will report the following error:

```
queryd: stored member records 65535 bytes
```

  For example:

```
struct MyStruct {
    string<1312721> m1;
};
string<1312721> 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);
strcpy(str, "131072");
m2[131072] = str;
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 Bug # 13782]

5 Custom Supported Platforms

Table 5-1 lists additional target libraries available with *RTI Data Distribution Service* 4.5x for which RTI offers custom support. If you are interested in using one of these platforms, please contact your local RTI representative or email **sales@rti.com**.

Table 5-1 **Custom Supported Platforms**

Operating System		CPU	Compiler	RTI Architecture Abbreviation
INTEGRITY	INTEGRITY 5.0.11	PPC970	GHnet3 TCT/IP stack	ppc970Linux5.0.11Gnu4.0.0
Linux	Mizbel Linux Kernel 2.6.32	ARM v7	binutils C++ Lite 2009q3-67 gcc 4.4.1	xenv7lnLinux2.6gcc4.4.1
	Red Hat Enterprise Linux 5.1	x86	gcc 3.4.6	i86Linux2.6gcc3.4.6
			Sun Java Platform Standard Edition JDK 1.5 and 1.6	i86Linux2.6gcc3.4.6jdk
	Red Hat Enterprise Linux 5.2 (Lenovell)	Pentium class	gcc 4.2.3	i86Linux2.6gcc4.2.3
			Sun Java Platform Standard Edition JDK 1.6	i86Linux2.6gcc4.2.3jdk
	Red Hat Enterprise Linux 5.2 (Novell SUSE Linux Enterprise Server 10 SP2 (2.6.20.70-expand4)	POWER 7	gcc 4.4.4	power7Linux2.6gcc4.4.4
			gcc 4.1.2	i86Hal64bit5.1gcc4.1.2
	Red Hat Enterprise Linux 5.1	x86	Sun Java Platform Standard Edition JDK 1.5 and 1.6	i86Hal64bit5.1gcc4.1.2jdk
			gcc 4.2.1	i86Hal64bit5.1gcc4.2.1
	Red Hat Enterprise Linux 5.4 (Lenovell)	Pentium class	Sun Java Platform Standard Edition JDK 1.6	i86Hal64bit5.1gcc4.2.1jdk