# *RTI Connext*

## Core Libraries and Utilities

## Release Notes

Version 4.5f

Your systems. Working as one.


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



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

# Contents

# Release Notes

This document includes the following sections:

- System Requirements (Section 1)
- Compatibility (Section 2)
- What's Fixed in Connext 4.5f (Section 3)
- Known Issues (Section 4)
- Custom Supported Platforms (Section 5)
- Experimental Features (Section 6)

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

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

# 1 System Requirements

## 1.1 Supported Operating Systems

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

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

Table 1.0 lists the platforms available with *Connext* 4.5f.

Table 1.0 **Platforms Available with Connext 4.5f**

| Platform | Operating System | Reference |
|---|---|---|
| AIX® | AIX 5.3 | Table 1.1 on page 6 |
| INTEGRITY® | INTEGRITY 5.0.11, 10 | Table 1.2 on page 7 |
| Linux® (Cell BE™) | Linux kernel 2.6.32 | Table 1.3 on page 7 |
| Linux (Intel®) | CentOS 5.4, 5.5<br>Fedora™ 12 (2.6.32 kernel)<br>Fedora 12 (2.6.32 kernel) with gcc 4.5.1<br>Red Hat® Enterprise Linux 5.0-5.2, 5.4, 5.5, 6.0, 6.1<br>Red Hat Enterprise Linux 5.2 with Real-Time Extensions<br>SUSE® Linux Enterprise Server 10.1 and 11 (2.6 kernel)<br>Ubuntu® Server 10.04 (2.6 kernel) | Table 1.4 on page 7 |
| Linux (PowerPC®) | Freescale™ P2020RDB (2.6.32 kernel)<br>SELinux (2.6.32 kernel)<br>Wind River® Linux 3<br>Yellow Dog™ Linux 4.0 | Table 1.5 on page 8 |
| LynxOS® | LynxOS 4.0, 4.2, 5.0 | Table 1.6 on page 9 |
| Mac OS® | Mac OS X 10.6 | Table 1.7 on page 9 |
| QNX® | QNX Neutrino® 6.4.1, 6.5 | Table 1.8 on page 9 |
| Solaris™ | Solaris 2.9, 2.10 | Table 1.9 on page 9 |

Table 1.0 **Platforms Available with Connext 4.5f**

| Platform | Operating System | Reference |
|---|---|---|
| VxWorks® | VxWorks 5.5, 6.3 - 6.9 | Table 1.10 on page 10 |
| VxWorks 653 | VxWorks 653 2.3 | |
| VxWorks MILS | VxWorks MILS 2.1.1 | |
| Windows® | Windows 7<br>Windows 2000 with service pack 2 or higher<br>Windows 2003 and Windows 2003 x64 Edition<br>Windows Vista®<br>Windows Server 2008 R2<br>Windows XP Professional<br>Windows XP Professional x64 Edition | Table 1.11 on page 13 |

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

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

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

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

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

### Visual Studio 2008 - Service Pack 1 Requirement

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

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

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

The Microsoft Visual C++ 2008 Service Pack 1 Redistribution Package can be obtained from the following Microsoft website:

- For x86 architectures:

  http://www.microsoft.com/downloads/details.aspx?familyid=A5C84275-3B97-4AB7-A40D-3802B2AF5FC2&displaylang=en

- For x64 architectures:

  http://www.microsoft.com/downloads/details.aspx?FamilyID=ba9257ca-337f-4b40-8c14-157cfdffee4e&displaylang=en

### Visual Studio 2010 - Service Pack 1 Requirement

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

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

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

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

- For x86 architectures:

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

- For x64 architectures:

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

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

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



## 1.2 Platforms Removed from Connext 4.5f

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

| Operating System | CPU | Compiler | RTI Architecture Abbreviation |
|---|---|---|---|
| Red Hat Enterprise Linux 4.0 (2.6 kernel) | x86 | gcc 3.4.3 | i86Linux2.6gcc3.4.3 |
| | | Sun Java Platform Standard Edition JDK 1.5 and 1.6 | i86Linux2.6gcc3.4.3jdk |
| | x64 | gcc 3.4.5 | x64Linux2.6gcc3.4.5 |
| | | Sun Java Platform Standard Edition JDK 1.5 and 1.6 | x64Linux2.6gcc3.4.5jdk |

## 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

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

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

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

- ❏ Linux/Fedora: all platforms except SELinux (2.6.32. kernel), Freescale P2020RDB (2.6.32. kernel), and Wind River Linux 3
- ❏ Solaris: all platforms
- ❏ VxWorks 6.7 and 6.8 (ppc604Vx6.7gcc4.1.2)
- ❏ Windows: all platforms

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

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

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

Table 1.1 **AIX Platforms**

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

Table 1.2 **INTEGRITY Platforms**

| Operating System | CPU | Compiler | IP Stack | RTI Architecture Abbreviation |
|---|---|---|---|---|
| INTEGRITY 5.0.11 | PPC 85xx | multi 4.2.4 | GHnet2 IP stack[1] | ppc85xxInty5.0.11.xes-p2020 |
| INTEGRITY 10.0.0 | x86 | multi 5.0.6 | CHNet IPv4 stack | pentiumInty10.0.0.pcx86 |

1. Kernel must be built using -lip4 or -lip46.

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

| Operating System | CPU | Compiler | RTI Architecture Abbreviation |
|---|---|---|---|
| Linux (2.6.32 kernel) | Cell BE | gcc 4.5.1, glib 2.9 | cell64Linux2.6gcc4.5.1 |

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 | i86Linux2.6gcc4.1.2 |
| | | Sun Java Platform Standard Edition JDK 1.6 | i86Linux2.6gcc4.1.2jdk |
| | x64 | gcc 4.1.2 | x64Linux2.6gcc4.1.2 |
| | | Sun Java Platform Standard Edition JDK 1.6 | x64Linux2.6gcc4.1.2jdk |
| 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 | x64Linux2.6gcc4.5.1 |
| Red Hat Enterprise Linux 5.0 (2.6 kernel) | x86 | gcc 4.1.1 | i86Linux2.6gcc4.1.1 |
| | | Sun Java Platform Standard Edition JDK 1.5 and 1.6 | i86Linux2.6gcc4.1.1jdk |
| | x64 | gcc 4.1.1 | x64Linux2.6gcc4.1.1 |
| | | Sun Java Platform Standard Edition JDK 1.5 and 1.6 | x64Linux2.6gcc4.1.1jdk |
| Red Hat Enterprise Linux 5.1, 5.2, 5.4, 5.5 (2.6 kernel) | x86 | gcc 4.1.2 | i86Linux2.6gcc4.1.2 |
| | | Sun Java Platform Standard Edition JDK 1.6 | i86Linux2.6gcc4.1.2jdk |
| | x64 | gcc 4.1.2 | x64Linux2.6gcc4.1.2 |
| | | Sun Java Platform Standard Edition JDK 1.6 | x64Linux2.6gcc4.1.2jdk |

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

| Operating System | CPU | Compiler | RTI Architecture Abbreviation |
|---|---|---|---|
| Red Hat Enterprise Linux 5.2 with Real-Time Extensions (2.6 kernel) | x86 | gcc 4.1.2 | i86Linux2.6gcc4.1.2 |
| | | Sun Java Platform Standard Edition JDK 1.5 and 1.6 | i86Linux2.6gcc4.1.2jdk |
| Red Hat Enterprise Linux 6.0, 6.1 (2.6 kernel) | x86 | gcc 4.4.5 | i86Linux2.6gcc4.4.5 |
| | x64 | gcc 4.4.5 | x64Linux2.6gcc4.4.5 |
| SUSE Linux Enterprise Server 10.1 (2.6 kernel) | x86 | gcc 4.1.0 | i86Suse10.1gcc4.1.0 |
| | | Sun Java Platform Standard Edition JDK 1.5 and 1.6 | i86Suse10.1gcc4.1.0jdk |
| | x64 | gcc 4.1.0 | x64Suse10.1gcc4.1.0 |
| | | Sun Java Platform Standard Edition JDK 1.5 and 1.6 | x64Suse10.1gcc4.1.0jdk |
| SUSE Linux Enterprise 11 Server Service Pack 1 (2.6 kernel) | AMD64 | gcc 4.3.4 | x64Linux2.6gcc4.3.4 |
| | | Sun Java Platform Standard Edition JDK 1.6 | x64Linux2.6gcc4.3.4jdk |
| Ubuntu Server 10.04 (2.6 kernel) | x86 | gcc 4.4.3 | i86Linux2.6gcc4.4.3 |
| | | Sun Java Platform Standard Edition JDK 1.5 and 1.6 | i86Linux2.6gcc4.4.3jdk |
| | x64 | gcc 4.4.3 | x64Linux2.6gcc4.4.3 |
| | | Sun Java Platform Standard Edition JDK 1.5 and 1.6 | x64Linux2.6gcc4.4.3jdk |

Table 1.5 **Linux Platforms on PowerPC CPUs**

| Operating System | CPU | Compiler | RTI Architecture Abbreviation |
|---|---|---|---|
| SELinux (2.6.32 kernel) | PowerPC440EP | gcc 4.5.1, glibc 2.9 | ppc4xxFPLinux2.6gcc4.5.1 |
| Yellow Dog® Linux 4.0 (2.6 kernel) (target only) | PPC 74xx (such as 7410) | gcc 3.3.3 | ppc7400Linux2.6gcc3.3.3 |
| Freescale P2020RDB (2.6.32 kernel) | PPC 85xx | Freescale gcc.4.3.74 based on gcc.4.3.2 | ppc85xxLinux2.6gcc4.3.2 |
| Wind River Linux 3 | PPC 85xx | gcc 4.3.2 | ppc85xxWRLinux2.6gcc4.3.2 |

Table 1.6 **LynxOS Platforms**

| Operating System | CPU | Compiler | RTI Architecture Abbreviation |
|---|---|---|---|
| LynxOS 4.0 | x86 | gcc 3.2.2 | i86Lynx4.0.0gcc3.2.2 |
| | | Sun Java Platform Standard Edition JDK 1.4 | i86Lynx4.0.0gcc3.2.2jdk |
| | PPC 74xx (such as 7410) | gcc 3.2.2 | ppc7400Lynx4.0.0gcc3.2.2 |
| | | Sun Java Platform Standard Edition JDK 1.4 | ppc7400Lynx4.0.0gcc3.2.2jdk |
| 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 |
| | | Sun Java Platform Standard Edition JDK 1.4 | ppc7400Lynx5.0.0gcc3.4.3jdk |

Table 1.7 **Mac OS Platforms**

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

Table 1.8 **QNX Platforms**

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

Table 1.9 **Solaris Platforms**

| Operating System | CPU | Compiler | RTI Architecture Abbreviation |
|---|---|---|---|
| Solaris 2.9 | x86 | gcc 3.3.2 | i86Sol2.9gcc3.3.2 |
| | | Sun Java Platform Standard Edition JDK 1.5 and 1.6 | i86Sol2.9jdk |
| | UltraSPARC | CC 5.4 (Forte Dev 7, Sun One Studio 7) | sparcSol2.9cc5.4 |
| | | gcc 3.2 | sparcSol2.9gcc3.2 |
| | | Sun Java Platform Standard Edition JDK 1.5 and 1.6 | sparcSol2.9jdk |

Table 1.9 **Solaris Platforms**

| Operating System | CPU | Compiler | RTI Architecture Abbreviation |
|---|---|---|---|
| Solaris 2.10 | AMD64 | gcc 3.4.3 | x64Sol2.10gcc3.4.3 |
| | | Sun Java Platform Standard Edition JDK 1.5 or 1.6 | x64Sol2.10jdk |
| | x86 | gcc 3.4.4 | i86Sol2.10gcc3.4.4 |
| | | Sun Java Platform Standard Edition JDK 1.5 or 1.6 | i86Sol2.10jdk |
| | UltraSPARC | gcc3.4.2 | sparcSol2.10gcc3.4.2 |
| | | Sun Java Platform Standard Edition JDK 1.5 or 1.6 | sparcSol2.10jdk |
| | UltraSPARC (with native 64-bit support) | gcc3.4.2 | sparc64Sol2.10gcc3.4.2 |
| | | Sun Java Platform Standard Edition JDK 1.5 or 1.6 | sparc64Sol2.10jdk |

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

| Operating System | CPU | Compiler | RTI Architecture |
|---|---|---|---|
| VxWorks 5.5 | PPC 603 | gcc 2.96 | ppc603Vx5.5gcc |
| | PPC 604 | gcc 2.96 | ppc604Vx5.5gcc |
| | PPC 750 | gcc 2.96 | ppc603Vx5.5gcc |
| | PPC 7400 | gcc 2.96 | ppc603Vx5.5gcc |
| VxWorks 6.3, 6.4 | Any Wind River PPC32 CPU with floating point hardware | gcc 3.4.4 | For kernel modules: ppc604Vx6.3gcc3.4.4<br>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<br>For Real Time Processes: ppc604Vx6.5gcc3.4.4_rtp |

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

| Operating System | CPU | Compiler | RTI Architecture |
|---|---|---|---|
| VxWorks 6.6 | Pentium | gcc 4.1.2 | For Kernel Modules:<br>pentiumVx6.6gcc4.1.2<br>For Real Time Processes:<br>pentiumVx6.6gcc4.1.2_rtp |
| | Any Wind River PPC32 CPU with floating point hardware | gcc 4.1.2 | For Kernel Modules:<br>ppc604Vx6.6gcc4.1.2<br>For Real Time Processes:<br>ppc604Vx6.6gcc4.1.2_rtp |
| | PPC 405[2] | gcc 4.1.2 | For Kernel Modules:<br>ppc405Vx6.6gcc4.1.2<br>For Real Time Processes:<br>ppc405Vx6.6gcc4.1.2_rtp |
| VxWorks 6.7 | Pentium | gcc 4.1.2 | For Kernel Modules:<br>pentiumVx6.7gcc4.1.2<br>For Real Time Processes:<br>pentiumVx6.7gcc4.1.2_rtp |
| | Any Wind River PPC32 CPU with floating point hardware | gcc 4.1.2 | For Kernel Modules:<br>ppc604Vx6.7gcc4.1.2<br>For Real Time Processes<br>on non-SMP systems:<br>ppc604Vx6.7gcc4.1.2_rtp<br>For Real Time Processes<br>on SMP systems:<br>ppc604Vx6.7gcc4.1.2_smp |
| | PPC 405[2] | gcc 4.1.2 | For Kernel Modules:<br>ppc405Vx6.7gcc4.1.2<br>For Real Time Processes:<br>ppc405Vx6.7gcc4.1.2_rtp |

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

| Operating System | CPU | Compiler | RTI Architecture |
|---|---|---|---|
| VxWorks 6.8 | Pentium | gcc 4.1.2 | For Kernel Modules: pentiumVx6.8gcc4.1.2<br>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<br>For Real Time Processes on a non-SMP system: ppc604Vx6.8gcc4.1.2_rtp |
| VxWorks 6.9 | Pentium32-bit | gcc 4.3.3 | For Kernel Modules: pentiumVx6.9gcc4.3.3<br>For Real Time Processes: pentiumVx6.9gcc4.3.3_rtp |
| | Pentium 64-bit | gcc 4.3.3 | For Kernel Modules: pentium64Vx6.9gcc4.3.3<br>For Real Time Processes: pentium64Vx6.9gcc4.3.3_rtp |
| | Any Wind River PPC32 CPU with floating point hardware | gcc 4.3.3 | For Kernel Modules: ppc604Vx6.9gcc4.3.3<br>For Real Time Processes: ppc604Vx6.9gcc4.3.3_rtp |
| VxWorks 653 2.3 | sbc8641d | gcc 3.32 | sbc8641Vx653-2.3gcc3.3.2 |
| | SIMPC | gcc 3.32 | simpcVx653-2.3gcc3.3.2 |
| VxWorks MILS 2.1.1 with vThreads 2.2.2 (VxWorks 5.5.1) Guest OS | ppc604 | gcc 3.3.2 | ppc604VxT2.2.2gcc3.3.2 |

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

Table 1.11 **Windows Platforms**

| Operating System | CPU | Compiler or Software Development Kit [1] [2] | RTI Architecture |
|---|---|---|---|
| Windows 7 | x86 | Visual Studio 2010 | i86Win32VS2010 |
| | | Visual Studio 2010 (C++/CLI, C# 8.0 or 9.0) | i86Win32dotnet4.0 |
| | | Sun Java Platform Standard Edition JDK 1.5 or 1.6 | i86Win32jdk |
| Windows 7 x64 Edition | x64 | Visual Studio 2010 | x64Win64VS2010 |
| | | Visual Studio 2010 (C++/CLI, C# 8.0 or 9.0) | x64Win64dotnet4.0 |
| | | Sun Java Platform Standard Edition JDK 1.5 or 1.6 | x64Win64jdk |
| Windows 2000 | x86 | Visual Studio 2005 SP1 | i86Win32VS2005 |
| | | Visual Studio 2005 SP 1 (C++/CLI, C# 8.0 or 9.0) | i86Win32dotnet2.0 |
| | | Sun Java Platform Standard Edition JDK 1.5 or 1.6 | i86Win32jdk |
| Windows 2003 | 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 SP1 | i86Win32VS2008 |
| | | Visual Studio 2008 SP 1 (C++/CLI, C# 8.0 or 9.0) | i86Win32dotnet2.0 |
| | | Sun Java Platform Standard Edition JDK 1.5 or 1.6 | i86Win32jdk |
| Windows 2003 x64 Edition | x64 | Visual Studio 2005 SP 1 | x64Win64VS2005 |
| | | Visual Studio 2005 SP 1 (C++/CLI, C# 8.0 or 9.0) | x64Win64dotnet2.0 |
| | | Visual Studio 2008 SP 1 | x64Win64VS2008 |
| | | Sun Java Platform Standard Edition JDK 1.5 or 1.6 | x64Win64jdk |
| Windows Server 2008 R2 x64 Edition | x64 | Visual Studio 2005 SP 1 (C++/CLI, C# 8.0 or 9.0) | x64Win64dotnet2.0 |
| | | Visual Studio 2010 | x64Win64VS2010 |
| | | Visual Studio 2010 (C++/CLI, C# 8.0 or 9.0) | x64Win64dotnet4.0 |
| | | Sun Java Platform Standard Edition JDK 1.5 or 1.6 | x64Win64jdk |
| Windows Vista | x86 | Visual Studio 2005 SP 1 | i86Win32VS2005 |
| | | Visual Studio 2005 SP 1 (C++/CLI, C# 8.0 or 9.0) | i86Win32dotnet2.0 |
| | | Visual Studio 2008 SP 1 | i86Win32VS2008 |
| | | Visual Studio 2008 SP 1 (C++/CLI, C# 8.0 or 9.0) | i86Win32dotnet2.0 |
| | | Sun Java Platform Standard Edition JDK 1.5 or 1.6 | i86Win32jdk |

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 SP 1 | x64Win64VS2005 |
| | | Visual Studio 2005 SP 1 (C++/CLI, C# 8.0 or 9.0) | x64Win64dotnet2.0 |
| | | Visual Studio 2008 SP1 | x64Win64VS2008 |
| | | Visual Studio 2008 SP 1 (C++/CLI, C# 8.0 or 9.0) | x64Win32dotnet2.0 |
| | | Sun Java Platform Standard Edition JDK 1.5 or 1.6 | x64Win64jdk |
| Windows XP Professional [3] | x86 | Visual Studio 2005 SP 1 | i86Win32VS2005 |
| | | Visual Studio 2005 SP 1 (C++/CLI, C# 8.0 or 9.0) | i86Win32dotnet2.0 |
| | | Visual Studio 2008 SP 1 | i86Win32VS2008 |
| | | Visual Studio 2008 SP 1 (C++/CLI, C# 8.0 or 9.0) | i86Win32dotnet2.0 |
| | | Sun Java Platform Standard Edition JDK 1.5 or 1.6 | i86Win32jdk |
| Windows XP Professional x64 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 |
| | | Visual Studio 2008 SP 1 (C++/CLI, C# 8.0 or 9.0) | x64Win32dotnet2.0 |
| | | Sun Java Platform Standard Edition JDK 1.5 or 1.6 | x64Win64jdk |

1. On Windows XP: If you are using JDK 5.0 and want to use Intel's HyperThreading technology, use JDK 5.0 Update 6 (build 1.5.0_06), which includes fixes to JNI and HyperThreading. (If you must use Update 5 (build 1.5.0_05), you should disable HyperThreading.)
2. The RTI .Net assemblies are supported for both the C++/CLI and C# languages. The type support code generated by rtiddsgen is in C++/CLI; compiling the generated type support code requires Microsoft Visual C++. Calling the assembly from C# requires Microsoft Visual C#.
3. Windows XP does not support IP_TOS unless registry changes are made. See http://support.microsoft.com/kb/248611, http://www.microsoft.com/technet/technetmag/issues/2007/02/CableGuy/default.aspx.

# 2 Compatibility

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

- Wire Protocol Compatibility (Section 2.1)
- Code Compatibility (Section 2.2)
- Data-Format Compatibility and Extensibility (Section 2.3)
- ODBC Database Compatibility (Section 2.4)
- Transport Compatibility (Section 2.5)

## 2.1 Wire Protocol Compatibility

### 2.1.1 General Information on RTPS (All Releases)

*Connext* communicates over the wire using 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 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 4.5

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

#### 2.1.2.1 RTPS Versions

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

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

If your *Connext* 4.5 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*.

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

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

This issue is resolved in *Connext* 4.5 and in *RTI Data Distribution Service* 4.4d-4.5e. As a result, by default, large data in *Connext* 4.5 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 4.5

#### 2.2.2.1 Type Support and Generated Code Compatibility

- **long long Native Data Type Support**

  In *Connext* 4.5 (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* 4.5 (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 APIs for the user-defined data type, resulting in incorrect behavior in the **lookup_instance()** and **get_key()** methods when using keyed data-types to communicate between applications in these languages and other programming languages. This issue was resolved in Java starting in *RTI Data Distribution Service* 4.3e rev. 01 and starting in .Net in *RTI Data Distribution Service* 4.4b.

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

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

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

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

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

- Samples with the same key may be identified as different instances. (For the case in which the *DataWriter* uses *RTI Data Distribution Service* 4.4d-4.5e or *Connext* 4.5, 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.

#### 2.2.2.2 Other API and Behavior Changes

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

If you are upgrading 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 avail-
able with -std=c++0x or -std=gnu++0x
MyFile.cpp:1060: warning: extended initializer lists only avail-
able 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);
```

This change is related to the fix described in Section 3.6.2.

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

Starting with *RTI Data Distribution Service* 4.5c (and thereby included in *Connext* 4.5), 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* 4.5), all the 'count' data types in DataReaderCacheStatus, DataReaderProtocolStatus, DataWriterCacheStatus and DataWriterProtocolStatus changed from 'long' to 'long long' in the C, C++ and .Net 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* 4.5), 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* 4.5), by default, the middleware is less restrictive (compared to older releases) on the writer side with regards to timestamps between consecutive samples: if the timestamp of the current sample is less than the timestamp of the previous sample by a small tolerance amount, **write()** will succeed.

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

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

Starting with *RTI Data Distribution Service* 4.4b (and thereby included in *Connext* 4.5), 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* 4.5) the default XML QoS Profiles file has been renamed and is installed in a new directory:

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

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

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

Starting with *RTI Data Distribution Service* 4.4d (and thereby included in *Connext* 4.5), 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 *RTI Data Distribution Service* 4.5c (and thereby included in *Connext* 4.5), 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* 4.5), the send window size of a *DataWriter* is set to infinite by default. This is done by changing the default values of two fields in DDS_RtpsReliableWriterProtocol_t (**min_send_window_size, max_send_window_size**) to be DDS_LENGTH_UNLIMITED.

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

## 2.3 Data-Format Compatibility and Extensibility

With *Connext*, 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 run time 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 new fields to it later without breaking existing components that may be unable to fill in those 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 feature set. Therefore, there is no guarantee that the persistent durability features will work with an arbitrary ODBC driver.

We have tested the following driver:

❏ MySQL ODBC 5.1.44

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

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

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

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

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

## 2.5 Transport Compatibility

The shared memory transport in *Connext* 4.5f does not interoperate with the shared memory transport in previous releases of *RTI Data Distribution Service*.

If two applications, one using *Connext* 4.5f and one using a previous release of *RTI Data Distribution Service*, run on the same node and they have the shared memory transport enabled, they will fail with the following error:

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

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

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

# 3 What's Fixed in Connext 4.5f

This section describes bugs fixed since the release of *RTI Data Distribution Service* 4.5e (the predecessor to *Connext* 4.5f).

## 3.1 Fixes Related to Reliability Protocol

### 3.1.1 Best Effort DataReaders Reported Lost Samples Filtered on Matching DataWriters

A *DataReader* using a ContentFilteredTopic and configured with best effort reliability was reporting the samples that were filtered by the matching *DataWriters* as lost.

This bug only occurred when the *DataWriter* was configured for writer-side filtering by setting **writer_qos.writer_resource_limits.max_remote_reader_filters** to a value greater than zero.

[RTI Bug # 14297]

### 3.1.2 Reliable DataReader May Have Stopped Receiving Samples After Receiving Out-of-Order Samples

In the previous release, when a reliable *DataReader* with a heartbeat response delay of zero received a sample out of order, depending on the subsequent number of out-of-order samples it received, it may have ended up in a state where it sent incorrect NACK messages that did not request any samples. Consequently, the *DataReader* may have been delayed or may have stopped receiving samples. This problem has been resolved.

[RTI Bug # 14268]

### 3.1.3 Reliable DataReader Stopped Receiving Samples from DataWriters with Batching and KEEP_LAST History

In the previous release, it was possible for a reliable *DataWriter* with batching enabled to write in a pattern such that large GAPs were formed between consecutive samples.

For example, consider a DataWriter with the History QoS kind set to KEEP_LAST with a small depth value, writing an instance A at the start, then only writing instance B later. This would introduce a large gap between the historical samples of A and B. When a late-joining *DataReader* received the historical samples, the GAP may have been larger than the *DataReader's* resource limits (**max_samples**, **max_samples_per_remote_writer**). When this occurred, the *DataReader* stopped receiving samples from the *DataWriter*.

[RTI Bug # 14278]

### 3.1.4 Writer Sends Redundant Repair Samples When NACK Response Is Delayed

A reliable *DataWriter* with delayed responses to a *DataReader's* requests for data (i.e. NACK response delays are non-zero) did not properly merge multiple requests received during the delay into a single repair packet. Consequently, the *DataWriter* may have been inefficient and sent redundant repair packets.

[RTI Bug # 14288]

## 3.2 Fixes Related to rtiddsgen

### 3.2.1 rtiddsgen Generates Corrupted VS Project Files in Japanese Environment

In a Japanese environment, *rtiddsgen* generated invalid characters in the <*type_name*>_publisher/subscriber-vs2010.vcxproj files. Visual Studio could not open the files and would report an error. This problem has been resolved.

[RTI Bug # 13896]

### 3.2.2 64-Bit rtiddsgen Visual Studio Project for .Net Referred to 32-Bit Connext DLLs

The 64-bit Visual Studio template for .Net mistakenly referred to the 32-bit *Connext* DLLs. This problem has been resolved.

[RTI Bug # 14328]

### 3.2.3 Error Processing Included Files when Using rtiddsgen -convertToXsd Option—AIX Platforms Only

If *rtiddsgen's* **-convertToXsd** option was used while compiling an IDL file that included another IDL file, *rtiddsgen* failed with an error such as:

```
Included file 'MyIncludeTypes.idl' not found
```

This error, which only occurred on AIX platforms, has been resolved.

[RTI Bug # 14223]

### 3.2.4 Example USER_QOS_PROFILES.xml not Generated by rtiddsgen for x64Suse10.1gcc4.1.0

*rtiddsgen* did not generate example USER_QOS_PROFILES.xml for the x64Suse10.1gcc4.1.0 architecture. This problem has been resolved.

[RTI # Bug 14260]

### 3.2.5 Java Compilation Error in Generated Code with Large Enums

If an IDL file contained an enum declaration with a large number of enumerators, the generated code in Java would not compile. You would have seen an error message like this:

```
MyEnumTypeCode.java: code too large
```

This problem has been resolved.

[RTI Bug # 14327]

## 3.3 Fixes Related to Transports

### 3.3.1 Estimate of Message Overhead For Large Data may have been Exceeded

The message overhead of samples with large data payloads (on the order of 64 KB) used to be underestimated. If not properly compensated by the transport's **message_size_max** property, the send operation would fail. To work around this issue, you may have set **message_size_max** to account for this overhead. This is no longer needed, as this issue has been resolved by adding a new property to the transport called **protocol_overhead_max**. See the *What's New* document for more details.

[RTI Bug # 11995]

### 3.3.2 Shutdown Hung in Face of Firewall and dds.transport.UDPv4.builtin.reuse_multicast_receive_resource—Linux Systems Only

For a Linux application running with the Linux firewall enabled, the application may have hung during *DomainParticipant* shutdown if the UDPv4 transport property, **reuse_multicast_receive_resource**, was disabled (its default value). To prevent this problem from happening out-of-the-box, the default value of this property has been changed to be enabled. The shutdown problem, however, still exists: the current work-around is to enable the UDPv4 property, **reuse_multicast_receive_resource**, which may have a small impact on performance by sending redundant multicast packets.

[RTI Bug # 14221]

### 3.3.3 Possible Message Corruption when Using Shared Memory Transport

In previous releases, if an application used the shared memory transport (enabled by default), some of the messages sent over shared memory may have been corrupted. Consequently, *DataReaders* may have reported errors when trying to deserialize the corrupted messages.

This problem has been resolved; however, the fix is not interoperable with previous versions of the shared memory transport. See Transport Compatibility (Section 2.5) for additional details.

### 3.3.4 Multicast Use on Interfaces with Multiple IP Addresses

On network interfaces with more than one IP address (for example, an IPv6 interface with a link-local IPv6 address and a SLAAC assigned IPv6 address), the RTI transports did not support joining multicast groups, and participant creation could fail if multicast was enabled. This problem has been resolved.

[RTI Bug # 13520]

### 3.3.5 Missing Transport Properties for UDPv4—Java API Only

In the Java API, the UDPv4 transport property structure was missing the following properties:

- **reuse_multicast_receive_resource**
- **interface_poll_period**
- **send_ping**
- **ignore_nonup_interfaces**

[RTI Bug # 14161]

### 3.3.6 Discovery Problems When Using TCP Transport in Asymmetric Mode

If there were multiple clients' *DomainParticipants* on the same node setting the TCP transport property **server_bind_port** to 0, they would not discover that the reachable peer's *DomainParticipant* set **server_bind_port** to a value other than 0. Likewise, the reachable peer's *DomainParticipant* did not discover the unreachable peer's *DomainParticipants*.

[RTI Bug # 14228]

### 3.3.7 Possible Segmentation Fault when Shutting Down Applications using TCP Transport

Applications using the TCP transport may have issued a segmentation fault during a graceful shut down. This problem has been resolved.

[RTI Bug # 14312]

## 3.4 Fixes Related to ContentFilteredTopics

### 3.4.1 Creation of SQL ContentFilteredTopic Sometimes Failed for Large Types—Java API Only

The creation of SQL ContentFilteredTopics in Java would fail if the serialized size of the associated TypeCode was greater than 65KB. This problem has been resolved.

[RTI Bug # 14200]

## 3.5 Fixes Related to Batching

### 3.5.1 DataReaders Using GROUP_PRESENTATION access_scope May Have Lost Samples if DataWriters Use Batching

*DataReaders* whose PresentationQosPolicy's **access_scope** was set to DDS_GROUP_PRESENTATION_QOS may have lost samples if their matching *DataWriters* were using batching.

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

- **writer_qos.protocol.rtps_reliable_writer.virtual_heartbeat_period**
- **writer_qos.protocol.rtps_reliable_writer.samples_per_virtual_heartbeat**

[RTI Bug # 14157]

### 3.5.2 Unexpected TIMEOUT Write Error when using Batching and KEEP_LAST History

The write operation in a *DataWriter* configured with batching and a History QoS kind of KEEP_LAST may have failed with a RETCODE_TIMEOUT return code if there were not enough resources to guarantee the depth set in the History QoS (**history.depth**) number of samples per instance. This issue occurred when **resource_limits.max_samples** < (**resource_limits.max_instances** * **history.depth**). This problem has been resolved.

[RTI Bug # 14281]

### 3.5.3 Potential Out of Order Delivery in Subscribers using DDS_GROUP_PRESENTATION_QOS access_scope if DataWriters used Batching

*Connext* applications that called the **begin_access()** and **end_access()** operations in *Subscribers* configured with DDS_GROUP_PRESENTATION_QOS **access_scope** may have received samples out of order if the matching *DataWriters* were configured to use batching.

The issue was likely to occur if, in addition to the use of batching, the *DataWriters* were configured with the History QoS kind set to KEEP_LAST.

[RTI Bug # 14290]

## 3.6 Fixes Related to Dynamic data

### 3.6.1 DynamicData's set_<primitive>_array APIs Should Have Failed if buffer_max_size Exceeded

Calls to DDS_DynamicData_set_<primitive>_array APIs did not fail when the length of the input sequence exceeded the DynamicData property **DDS_DynamicDataProperty_t.buffer_max_size**. Consequently, if you sent the DynamicData sample with a *DataWriter*, the *DataReader* received a zero-length sequence.

[RTI Bug # 14035]

### 3.6.2 Constructor Missing for Some Dynamic Data Structures—C++ API Only

The following structures had to be initialized using a C-style initializer because they didn't define a C++ constructor:

- DDS_DynamicDataTypeProperty_t
- DDS_DynamicDataProperty_t
- DDS_DynamicDataTypeSerializationProperty_t

This problem has been resolved; a constructor without parameters that sets the default values is now included. Existing C++ applications that worked around this issue by using the a C-style initializer must be modified not to use it anymore; otherwise you will get a compilation error. See Section 2.2.2.2 for details on the required code change.

[RTI Bug # 14320]

## 3.7 Other Fixes

### 3.7.1 Possible Delay in Delivering Rejected Samples

If the resource limit, **max_samples_per_instance**, was exceeded, rejected samples were not always provided to the application after space becomes available on the *DataReader's* queue. Rejected samples were only provided to the application after space became available and new samples were received.

This problem has been resolved. Now rejected samples are provided to the application as soon as there is space for them on the *DataReader's* queue, regardless of the reception of new samples.

[RTI Bug # 14140]

### 3.7.2 Wait_for_historical_data() Could Return Prematurely

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

[RTI Bug # 14296]

### 3.7.3 DataWriter's get_qos() Assigned Incorrect Value to publication_name

The *DataWriter's* **get_qos()** operation returned an incorrect value for the **writer_qos.publication_name** field. This problem has been resolved.

[RTI Bug # 14253]

### 3.7.4 Waitset Awakens without Triggering Condition

It was possible for a Waitset to be awakened without any triggering condition being true, or having been true. In this release, false awakenings should be greatly reduced. Note that it is still possible for a Waitset to return without a triggering condition, but only if the triggering condition becomes true (awakening the waitset), and then reverts to false before the waitset returns.

[RTI Bug # 14081]

### 3.7.5 Potential Segmentation Fault on Keyed DataReaders using ContentFilteredTopic

Samples written by a *DataWriter* contains user data and additional metadata specific to the sample such as the key-hash or ContentFilteredTopic result. A rare problem with the size of the buffer used by a *DataWriter* to serialize these metadata may have caused the corresponding *DataReader(s)* to misinterpret the metadata and user data, and consequently caused the *DataReader(s)* to crash. This problem has been resolved.

[RTI Bug # 14257]

### 3.7.6 Possible Segmentation Fault when Receiving Discovery RTPS Vendor-Specific ParametersIds

An application may have crashed with a segmentation fault or access violation if it received a discovery RTPS DATA message containing a non-RTI vendor-specific ParameterId that collided with an RTI vendor-specific ParameterId. This issue has been resolved.

[RTI Bug # 14284]

### 3.7.7 Incorrect Result Returned by Post-increment Operator (++) in DDS::SequenceNumber

The post-increment operator acted as a pre-increment operator in previous releases due to an error in the implementation.

[RTI Bug # 14162]

### 3.7.8 Failure to Load Discovery Plugin did not Cause DomainParticipant Creation to Fail

If a discovery plugin failed to load (for example, due to a misspelled library name or an incorrect system library path), a *DomainParticipant* was still created. Only a warning message was logged. This problem has been resolved. Now if a discovery plugin cannot be loaded, the *DomainParticipant* will fail with an error message.

[RTI Bug # 14177]

### 3.7.9 Unused Arguments in dds_c_infrastructure.h

There were unused arguments in **dds_c_infrastructure.h**, which led to compiler warnings such as these:

```
In member function 'DDS_SequenceNumber_t DDS_SequenceNumber_t::opera-
tor++(int)':
dds_c_infrastructure.h:381:30: warning: variable 'result' set but not
used [-Wunused-but-set-variable]
dds_c_infrastructure.h: At global scope:
dds_c_infrastructure.h:380:33: warning: unused parameter 'unused'
[-Wunused-parameter]
dds_c_infrastructure.h:391:33: warning: unused parameter 'unused'
[-Wunused-parameter]
```

This problem has been resolved.

[RTI Bug # 14198]

### 3.7.10 Full GUID Structure not Shown by rtiddsspy

*rtiddsspy* only showed 12 of the 16 bytes of the GUID structure. This problem has been resolved.

[RTI Bug # 14339]

### 3.7.11 Possible Segmentation Fault if Instance Disposed/Unregistered and DataReader Queue Exceeded max_samples

If a *DataReader* had a finite **resource_limits.max_samples** and the *DataWriter* disposing or unregistering the instance was configured with **protocol.serialize_key_with_dispose** set to TRUE (not the default configuration), this may have caused a segmentation fault.

[RTI Bug # 13820]

### 3.7.12 DataReader May Report Incorrect Sample Lost Status

The DDS_SampleLostStatus value returned by the *DataReader's* function **get_sample_lost_status()** or the callback **on_sample_lost()** may have been incorrect, causing the *DataReader* may have reported report lost samples when there were none.

This only occurred if the *DataWriter* was configured with a finite **max_samples_per_instance**, its History QoS kind was KEEP_ALL, and the write operation blocked because the previous resource limit was hit for a given instance.

[RTI Bug # 14226]

### 3.7.13 Possible Segmentation Fault After Sample Losses

The *DataReader* may issued a segmentation fault if there were sample losses because the *DataWriter* removed samples from its queue before they were received by the *DataReader*. Although triggered by sample losses, this bug was not likely to occur unless the losses were combined with sample rejection as a result of exceeding the *DataReader's* **max_samples_per_instance** resource limit. This problem has been resolved.

[RTI Bug # 14227]

### 3.7.14 False Positives Returned by DDS_Time_zero()

The **DDS_Time_zero()** API returned true when sec *or* nanosec were zero. This problem has been resolved. Now it will only return true is both sec *and* nanosec are zero.

### 3.7.15 Wait_for_historical_data May Have Timed Out Incorrectly

This issue occurred in a late-joining reader situation in which the remote writer was no longer writing data after the reader joined the system and the historical data included GAPs. In this situation, it was possible for **wait_for_historical_data()** to time-out even when all the historical data had already been received. This problem has been resolved.

[RTI Bug # 14234]

### 3.7.16 Asynchronous Publisher Crash on Shutdown

Previously, when an asynchronous publishing *DataWriter* was deleted, there was a race condition that could have resulted in a crash of the *DataWriter*. This problem has been resolved.

[RTI Bug # 14322]

### 3.7.17 File Permissions Problem with Professional Edition Installer

In the previous version of the *Professional Edition* installer, some files were installed with incorrect permissions. This problem has been resolved.

[RTI Bug # 14210]

#### 3.7.18 Version Header File not Included in ndds_c.h

A new header file, **ndds_version.h**, was introduced in 4.5e. This file contains macros that define the version number of the release. Because **ndds_version.h** was mistakenly not included in **ndds_c.h**, user applications built with 4.5e had to manually include **ndds_version.h** in order to use the new macros. This issue has been resolved; now **ndds_c.h** includes **ndds_version.h**.

[RTI Bug # 14337]

# 4 Known Issues

## 4.1 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 # 10765]

## 4.2 Disabled Interfaces on Windows Systems

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

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

## 4.3 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 # 11362]

## 4.4 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 APIs, or CORBA-compatible C++ type plugins.

[RTI Bug # 12606]

## 4.5 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 Bug # 9014]

## 4.6 .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 Bug # 13088]

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

If your *Connext* 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 similar to the following in your *Connext* application:

```
PRESWriterHistoryDriver_initializeSample:!serialize
WriterHistoryMemoryPlugin_addEntryToSessions:!initialize sample
WriterHistoryMemoryPlugin_getEntry:!add virtual sample to sessions
WriterHistoryMemoryPlugin_addSample:!get entry
PRESWriterHistoryDriver_addWrite:!add_sample
PRESPsWriter_write:!collator addWrite
RTIDefaultMonitorParticipantObject_sampleAndPublishWriterDesc:!write
RTIDefaultMonitorPublisher_onEventNotify:!publish writer desc
```

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

```
rti.monitor.config.skip_writer_properties
rti.monitor.config.skip_reader_properties
rti.monitor.config.skip_participant_properties
```

[RTI Bug # 13625]

## 4.8 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 Bug # 14313]

## 4.9 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 Bug # 12647]

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

  [RTI 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 Bug # 12897]

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

  [RTI 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 Bug # 13745]

# 5 Custom Supported Platforms

Table 5.1 lists additional target libraries available with *Connext* 4.5f, for which RTI offers custom support. If you are interested in using one of these platforms, please contact your local RTI representative or email **sales@rti.com**.

Table 5.1 **Custom Supported Platforms**

| Operating System | | CPU | Compiler | RTI Architecture Abbreviation |
|---|---|---|---|---|
| INTEGRITY | INTEGRITY 5.0.11 | PPC8349 | GHnet2 TCP/IP stack | ppc8349Inty5.0.11.mds8349 |
| Linux | Mistral Linux Kernel 2.6.32 | ARMv7 | Sourcery G++ Lite 2009q3-67 gcc 4.4.1 | armv7leLinux2.6gcc4.4.1 |
| | Red Hat Enterprise Linux 5.1 | x86 | gcc3.4.6 | i86Linux2.6gcc3.4.6 |
| | | | Sun Java Platform Standard Edition JDK 1.5 and 1.6 | i86Linux2.6gcc3.4.6jdk |
| | Red Hat Enterprise Linux 5.2 (2.6 kernel) | Pentium class | gcc 4.2.1 | i86Linux2.6gcc4.2.1 |
| | | | Sun Java Platform Standard Edition JDK 1.6 | i86Linux2.6gcc4.2.1jdk |
| | Red Hat Enterprise Linux 6 for IBM POWER7 Servers (2.6.32-70.el.ppc64) | POWER 7 | gcc 4.4.4 | power7Linux2.6gcc4.4.4 |
| | RedHawk Linux 5.1 | x86 | gcc 4.1.2 | i86RedHawk5.1gcc4.1.2 |
| | | | Sun Java Platform Standard Edition JDK 1.5 and 1.6 | i86RedHawk5.1gcc4.1.2jdk |
| | RedHawk Linux 5.4 (2.6 kernel) | Pentium class | gcc 4.2.1 | i86RedHawk5.4gcc4.2.1 |
| | | | Sun Java Platform Standard Edition JDK 1.6 | i86RedHawk5.4gcc4.2.1jdk |
| | RedHawk Linux 6.0 | x64 | gcc 4.4.5 | x64Linux2.6gcc4.4.5 |

# 6 Experimental Features

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

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

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

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

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

### Disclaimers

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

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

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