RTI Connext DDS

Core Libraries and Utilities

What's New

in Version 5.1.0

Your systems. Working as one.

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

What's New

This document highlights new or changed features in *RTI*® *Connext*™ (formerly *RTI Data Distribution Service*) 5.1.0 compared to the previous release, 5.0.0. (For details on fixed bugs, please see the *RTI Core Libraries and Utilities Release Notes*.)

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

❑ Use the RTI Customer Portal (http://support.rti.com) to download RTI software, access documentation and contact RTI Support. The RTI Customer Portal requires a username and password. You will receive this in the email confirming your purchase. If you do not have this email, please contact **license@rti.com**. Resetting your login password can be done directly at the RTI Customer Portal.

❑ The RTI Community website (http://community.rti.com) provides a wealth of knowledge to help you use *RTI Connext DDS*, including:

- Best Practices,
- Example code for specific features, as well as more complete use-case examples,
- Solutions to common questions,
- A glossary,
- Downloads of experimental software,
- And more.

❑ Whitepapers and other articles are available from http://www.rti.com/resources.

1 Supported Platforms

1.1 New Standard Target Platforms

❑ This release adds support for the platforms described in Table 1.1. Please see the updated *RTI Core Libraries and Utilities Platform Notes* for details on using these platforms.

❑ When using VxWorks® 6.3 and higher with Real Time Processes (RTP mode):

Shared libraries are now available for MIPS CPUs. (Shared libraries are also available for Pentium CPUs but remain unavailable for PPC CPUs.)

1.2 New Custom Target Platforms

This release adds custom support for the platforms in Table 1.2. If you are interested in using one of these platforms, please contact your local RTI representative or email **sales@rti.com**.

Table 1.1 **New Platforms**

Operating System		CPU	Compiler	RTI Architecture Abbreviation
AIX	AIX® 7.1	POWER class (64-bit mode)	IBM® xlC_r for AIX v12.1	64p7AIX7.1xlc12.1
			IBM Java 1.7	64p7AIX7.1xlc12.1jdk
INTEGRITY	INTEGRITY® 10.0.2	p4080 (based on e500mc core)	Multi 6.1	p4080Integrity10.0.2.xes-p4080-smp[1]
Linux	CentOS 6.0, 6.2-6.4 (2.6 kernel)	x86	gcc 4.4.5	i86Linux2.6gcc4.4.5
			Java Platform, Standard Edition JDK 1.7	i86Linux2.6gcc4.4.5jdk
		x64	gcc 4.4.5	x64Linux2.6gcc4.4.5
			Java Platform, Standard Edition JDK 1.7	x64Linux2.6gcc4.4.5jdk
	Raspbian Wheezy 7.0 (3.x kernel)	ARMv6	gcc 4.7.2[2]	armv6vfphLinux3.xgcc4.7.2
			Java Platform, Standard Edition JDK 1.7	armv6vfphLinux3.xgcc4.7.2jdk
	Red Hat® Enterprise Linux® 4.0 (2.6 kernel)	x86	gcc 3.4.3	i86Linux2.6gcc3.4.3
		x64	gcc 3.4.5	x64Linux2.6gcc3.4.5
	Red Hat Enterprise Linux 6.2-6.4 (2.6 kernel)	x86	gcc 4.4.5	i86Linux2.6gcc4.4.5
			Java Platform, Standard Edition JDK 1.7	i86Linux2.6gcc4.4.5jdk
		x64	gcc 4.4.5	x64Linux2.6gcc4.4.5
			Java Platform, Standard Edition JDK 1.7	x64Linux2.6gcc4.4.5jdk
	SUSE® Linux Enterprise Server 11 SP2 (3.x kernel)	x86	gcc 4.3.4	i86Linux3gcc4.3.4
			Java Platform, Standard Edition JDK 1.7	i86Linux3gcc4.3.4jdk
	Ubuntu® Server 12.04 LTS	x86	gcc 4.6.3	i86Linux3.xgcc4.4.3
			Java Platform, Standard Edition JDK 1.7	i86Linux3.xgcc4.6.3jdk
		x64	gcc 4.6.3	x64Linux3.xgcc4.6.3
			Java Platform, Standard Edition JDK 1.7	x64Linux3.xgcc4.6.3jdk
Mac OS	Mac OS® X 10.8	x64	clang 4.1	x64Darwin12clang4.1
			Java Platform, Standard Edition JDK 1.7	x64Darwin12clang4.1jdk
VxWorks	VxWorks 6.9.3.2[3]	x64	gcc 4.3.3	For Kernel Modules: pentium64Vx6.9gcc4.3.3 For Real Time Processes: pentium64Vx6.9gcc4.3.3_rtp

Table 1.1 **New Platforms**

Operating System		CPU	Compiler	RTI Architecture Abbreviation
Windows	Windows® 8 (32-bit Edition)	x86	Visual Studio® 2012	x64Win64VS2012
			Visual Studio 2012 (C++/CLI, C# 8.0 or 9.0)	i86Win32dotnet4.5
			Java Platform, Standard Edition JDK 1.7	x64Win64jdk
	Windows 8 (64-bit Edition)	x64	Visual Studio 2012	x64Win64VS2012
			Visual Studio 2012 (C++/CLI, C# 8.0 or 9.0)	x64Win64dotnet4.5
			Java Platform, Standard Edition JDK 1.7	x64Win64jdk
	Windows Server 2012 (64-bit Edition)	x64	Visual Studio 2012	x64Win64VS2012
			Visual Studio 2012 (C++/CLI, C# 8.0 or 9.0)	x64Win64dotnet4.5
			Java Platform, Standard Edition JDK 1.7	x64Win64jdk

1. Only C and C++ APIs are supported.
2. Requires Linaro Gnueabihf Cross Compiler
3. Available Q1 2014

Table 1.2 **New Custom Target Libraries (CTLs)**

Operating System		CPU	Compiler	RTI Architecture Abbreviation
Linux	NI Linux 3.2	ARMv7	gcc 4.4.1	armv7AngstromLinux3.2gcc4.4.1.cortex-a9
VxWorks	VxWorks 6.7	Any PowerPC CPU with floating-point hardware that is backwards-compatible with 32-bit PowerPC 604[1]	JamaicaVM 6.2.1 with gcc 4.1.2	For Kernel Modules: ppc604Vx6.7gcc4.1.2jdk
	VxWorks 6.8			For Kernel Modules: ppc604Vx6.8gcc4.1.2jdk

1. Some PowerPC cores such as e500v1 and e500v2 are not fully backwards-compatible with PPC 604.

1.3 Removed Platforms

The platforms in Table 1.3 were supported in 5.0.0 but are not supported in this release.

Table 1.3 **Removed Platforms**

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

Table 1.3 **Removed Platforms**

Operating System		CPU	Compiler	RTI Architecture Abbreviation
Linux	Mistral Linux Kernel 2.6.32 (Custom Target Platform)	ARMv7	Sourcery G++ Lite 2009q3-67 gcc 4.4.1	armv7leLinux2.6gcc4.4.1
	Red Hat Enterprise Linux 5.1 (Custom Target Platform)	x86	gcc3.4.6	i86Linux2.6gcc3.4.6
			Sun Java Platform Standard Edition JDK 1.5 and 1.6	i86Linux2.6gcc3.4.6jdk
	RedHawk Linux 5.1 (Custom Target Platform)	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) (Custom Target Platform)	x86	gcc 4.2.1	i86RedHawk5.4gcc4.2.1
			Sun Java Platform Standard Edition JDK 1.6	i86RedHawk5.4gcc4.2.1jdk
	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
	Ubuntu Server 10.04 (LTS)	x86	gcc 4.4.3	i86Linux2.6gcc4.4.3
			Sun Java Platform Standard Edition JDK 1.6	i86Linux2.6gcc4.4.3jdk
		x64	gcc 4.4.3	x64Linux2.6gcc4.4.3
			Sun Java Platform Standard Edition JDK 1.6	x64Linux2.6gcc4.4.3jdk
Mac OS	Mac OS X 10.6	x64	gcc 4.2.1	x64Darwin10gcc4.2.1
			Java SE 1.6 for Mac OS	x64Darwin10gcc4.2.1jdk
Solaris	Solaris™ 2.9	x86	Sun Java Platform Standard Edition JDK 1.6	i86Sol2.9jdk
		UltraSPARC	Sun Java Platform Standard Edition JDK 1.6	sparcSol2.9jdk

2 Extensible Type Support

2.1 Support for Mutable Types

This release adds partial support in the C, C++, Java, and .NET APIs for mutable types, as defined in the "Extensible and Dynamic Topic Types for DDS" (DDS-XTypes) specification from the Object Management Group (OMG).

Using mutable types allows data types to evolve over time by adding, removing, or changing the order of the members in the type definitions. Specifically, these features are now supported:

❑ Specifying the MUTABLE_EXTENSIBILITY value for the Extensibility annotation in IDL, XML, and XSD. For example, to use this value in IDL:

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

- ❏ Specifying the member ID in IDL, XML, and XSD with the built-in annotation, ID. For example, to use the ID annotation in IDL:

```
struct MyStruct {
    long m1; //@ID 100
    long m2; //@ID 200
}; //@Extensibility MUTABLE_EXTENSIBILITY
```

- ❏ Accessing the member ID information in a TypeCode by using the **DDS_TypeCode_member_id()** operation.
- ❏ Propagating member ID information on the wire as part of the TypeObject.
- ❏ Accessing and setting the value of members of a mutable type using the member name in existing DynamicData API. Accessing or setting by member ID is not supported at this time.
- ❏ Using content filters for mutable types

For more information on using mutable types, please see the updated *RTI Core Libraries and Utilities Getting Started Guide Addendum for Extensible Types*.

2.2 Support for Optional Members

This release adds support for optional members in C, C++ and Java, as defined in the "Extensible and Dynamic Topic Types for DDS" (DDS-XTypes) specification from the Object Management Group (OMG).

In a structure type, an optional member is a member that an application can decide to send or omit as part of every published sample. Specifically, these features are now supported:

- ❏ Declaring struct members as optional in IDL, XML and XSD
- ❏ Generating code for types with optional members in C, C++ and Java
- ❏ Accessing and setting optional members in the existing DynamicData API using the member name. Accessing or setting by member ID is not supported at this time.
- ❏ Using content filters for types with optional members

For more information on using optional members, see the updated *RTI Connext Getting Started Guide Addendum for Extensible Types*.

2.3 More Information Printed if DataReader and DataWriter cannot Communicate due to Incompatible Types

Connext DDS checks that a *DataReader* and *DataWriter* for the same topic use types that are compatible. If a *DataReader's* TypeConsistencyEnforcementQosPolicy **kind** is set to DDS_DISALLOW_TYPE_COERCION, the middleware requires that both types are equal. In the previous release, the warning messages did not include the specific reason why the types were not equal. The warning messages now include more details. For example:

```
RTICdrTypeObjectStructureType_equals:types are not equal: different number
of members
RTICdrTypeObjectTypeLibraryElement_equals:types are not equal: TypeA, TypeB
```

Warning messages used when the TypeConsistencyEnforcementQosPolicy **kind** is DDS_ALLOW_TYPE_COERCION (the default) have also been improved.

3 Improvements to Ease-of-Use and Out-of-the-Box Experience

3.1 Built-in QoS Profiles

A number of QoS profiles are now built into the *Connext* core libraries and can be used as starting points when configuring QoS for your *Connext* applications.

There are two provided libraries, **BuiltinQosLib** and **BuiltinQosLibExp**, and 34 different profiles. You can use any of these profiles as base profiles when creating your own XML configurations or simply use these profiles directly in the **DDS_*_create_*_with_profile()** APIs.

There are three types of profiles:

- ❑ **Baseline.X.X.X** profiles represent the QoS defaults for *Connext* version X.X.X. The defaults for the latest *Connext* version can be accessed using the **BuiltinQosLib::Baseline** profile.
- ❑ **Generic.X** profiles allow you to easily configure different features and communication use-cases with *Connext*. For example, there is a **Generic.StrictReliable** profile for use when your application has a requirement for no data loss, regardless of the application domain.
- ❑ **Pattern.X** profiles inherit from **Generic.X** profiles and allow you to configure various domain-specific communication use cases. For example, there is a **Pattern.Alarm** profile that can be used to manage the generation and consumption of alarm events.

The **USER_QOS_PROFILES.xml** file generated by *rtiddsgen* now contains a profile that inherits from the **BuiltinQosLibExp::Generic.StrictReliable** profile as an example of how to use these profiles in your own application.

Example use cases for these profiles:

- ❑ To quickly enable *RTI Monitoring Library* by inheriting from the **BuiltinQosLib::Generic.Monitoring.Common** profile.
- ❑ To easily revert to the default QoS values from a previous *Connext* version by inheriting from the correct **BuiltinQosLib::Baseline.X.X.X profile.**
- ❑ To set up common use-case configurations and patterns such as strict reliability or large data communication by inheriting from one of the **BuiltinQosLibExp::Generic.X** or **Pattern.X** profiles.

To see the contents of the built-in QoS profiles:

In ***<Connext installation directory>*/resource/qos_profiles_5.1.0/xml**, you will find:

- ❑ **BaselineRoot.documentationONLY.xml**—This file contains the root baseline QoS profile corresponding to the default values of *Connext* 5.0.0.
- ❑ **BuiltinProfiles.documentationONLY.xml**—This file contains the rest of the built-in QoS profiles.

These profiles are also described in the API Reference HTML documentation (under **Modules/ RTI Connext DDS API Reference/Infrastructure Module/Builtin QoS Profiles**) and in Section 17.9.5, *Built-in QoS Profiles*, in the *Core Libraries and Utilities User's Manual*.

Notes:

- ❑ The built-in QoS profiles that enable RTI Monitoring Library set the property **rti.monitor.create_function**. Consequently, they only work in *Connext* applications in which the monitoring library can be loaded dynamically. Specifically, the built-in monitoring profiles will not work in these situations:

- When the *Connext* application links the monitoring libraries statically
- When using a VxWorks 6.7 or 6.8 platform with Java[1].

For more information, see the *RTI Monitoring Library Getting Started Guide* (**RTI_Monitoring_Library_GettingStarted.pdf**).

❏ Some of the built-in profiles are experimental. All the experimental profiles are contained within the library **BuiltinQosLibExp**.

3.2 New Default Transport Settings

Some of the default settings for a number of the provided transport plugins have changed to provide better out-of-the-box performance. By increasing the out-of-the-box performance, in most cases you will not have to modify these settings or keep them in sync across all *Connext* applications and services within your system.

The affected transports are UDPv4, UDPv6, Secure WAN, TCP, and shared memory.

For UDPv4, UDPv6, WAN, and TCP, we changed the values for **message_size_max**, **send_socket_buffer_size**, and **recv_socket_buffer_size**.

For shared memory, we changed the values for **message_size_max**, **received_message_count_max**, and **recv_buffer_size**.

Table 3.1 and Table 3.2 show the old and new default values.

Table 3.1 **UDPv4, UDPv6, WAN, and TCP**

	Old Default (bytes)	New Default (bytes)	
		Non-INTEGRITY Platforms	INTEGRITY Platforms[1]
message_size_max	9216	65507[2]	9216
send_socket_buffer_size	9216	131072	131072
recv_socket_buffer_size	9216	131072	131072

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

Table 3.2 **Shared Memory**

	Old Default (bytes)	New Default (bytes)	
		Non-INTEGRITY Platforms	INTEGRITY Platforms[1]
message_size_max	9216	65536	9216
received_message_count_max	32	64	8
receive_buffer_size	73728	1048576	18432

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

1. VxWorks 6.7 and 6.8 Java platforms require custom supported libraries.

Important Notes:

- ❏ If you are using a LynxOS platform over the loopback interface, you may have to decrease the default value for **message_size_max** because (due to a LynxOS bug) the loopback interface does not allow IP fragmentation. Please see the *RTI Core Libraries and Utilities Platform Notes* for more information.
- ❏ Because the default value for **message_size_max** changed, *Connext* 5.1.0 is not out-of-the-box backwards compatible with older versions. For details, please see Section 2.5.2, Transport Compatibility for *Connext* 5.1.0 in the *RTI Core Libraries and Utilities Release Notes*.

3.3 New Automatic Value for ReceiverPoolQosPolicy's buffer_size

In previous releases, increasing the **message_size_max** of an installed transport usually required adjusting the ReceiverPoolQoSPolicy's **buffer_size**. The **buffer_size** must always be at least as large as the maximum **message_size_max** of any installed non-zero-copy transport.[1]

In this release, the **buffer_size** can be adjusted automatically by the middleware by configuring its value to DDS_LENGTH_AUTO (in C/C++) or ReceiverPoolQosPolicy.LENGTH_AUTO (in .NET and Java). When **buffer_size** is set to DDS_LENGTH_AUTO (the default value), the effective value will automatically be set to the largest **message_size_max** of all installed transports, without needing any other configuration. You should not need to change this value to anything other than the default.

3.4 'XML-Based Application Creation' Feature No Longer Considered Experimental

The XML-Based Application Creation feature, introduced in *Connext* 4.5f, is no longer considered 'experimental.' This feature simplifies the development and programming of *RTI Connext* applications by allowing you to define your *entire* system using XML.

Without this feature, you can define data types and Quality of Service settings in XML. With this feature, you can also use XML to define your system's *Topics*, *DomainParticipants*, and all the *Entities* they contain (*Publishers*, *Subscribers*, *DataWriters* and *DataReaders*).

For details on using this feature, see the *XML-Based Application Creation Getting Started Guide* (**RTI_CoreLibrariesAndUtilities_XML_AppCreation_GettingStarted.pdf**).

3.5 New APIs to Get ParticipantBuiltinTopicData of Matched Publication or Subscription using Publication and Subscription Handles

Two new APIs, **get_matched_publication_participant_data()** (for a *DataReader*) and **get_matched_subscription_participant_data()** (for a *DataWriter*), allow you to get the DDS_ParticipantBuiltinTopicData of a matched publication or subscription using publication and subscription handles. Previously, retrieving the DDS_ParticipantBuiltinTopicData was only possible using a corresponding participant handle. These APIs are safe to use in listener callbacks.

3.6 Equals Functions for QoS Objects Now Available for All Languages

All QoS objects have 'equals' functions that check two QoS objects for strict equality. In previous versions, this feature was only available in the Java API. Now all of supported language APIs support the QoS equals functions.

1. A "zero-copy transport" does not use the receive buffer. A transport is zero-copy if the properties_bitmap property in the DDS_Transport_Property_t is NDDS_TRANSPORT_PROPERTY_BIT_BUFFER_ALWAYS_LOANED. The only built-in transport that supports zero-copy is the UDPv4 transport on VxWorks platforms.

3.7 Automatic Participant Announcement to Default Domain Zero

Starting with this release, DomainParticipants announce themselves to default domain ID 0 using the default UDPv4 multicast group address (239.255.0.1) for discovery traffic on that domain.

This feature is used by tools such as *RTI Admin Console* and *RTI Monitor* to monitor and provide a list of all active domains in a system.

The feature is configured using two new fields in the DiscoveryConfigQosPolicy:

- **default_domain_announcement_period**
- **ignore_default_domain_announcements**

By default, participant announcements to the default domain are sent every 30 seconds. You can choose to disable this feature by setting **default_domain_announcement_period** to DDS_DURATION_INFINITE.

Notice that announcement to the default domain is orthogonal to regular discovery.

For additional information, see the *RTI Core Libraries and Utilities Users Manual*.

3.8 More Information Available in DomainParticipant's PropertyQosPolicy

In the previous release, new system information such as **hostname** and **process_id** were added to the DomainParticipant's PropertyQosPolicy. This releases adds even more information, such as:

- **username** — the username that is running the process
- **execution_timestamp**— the time when the execution started
- **creation_timestamp** — the time when the executable was created
- **executable_filepath** — the name and full path of the executable
- **target** — the architecture for which the library was compiled (for example, x64Darwin10gcc4.2.1)

The **username**, **execution_timestamp**, **creation_timestamp**, and **executable_filepath** properties are only supported on Windows and Linux architectures.

By default, these properties, like any other properties, are propagated. It is always possible to disable propagation by setting the property's **propagate** field to FALSE. For more information, see Section 8.7, System Properties, in the *RTI Core Libraries and Utilities User's Manual*.

3.9 Convenience Constructors and Constants for Duration_t and Time_t

The classes Duration_t and Time_t include four new convenience constructors: **from_micros()**, **from_millis()**, **from nanos()**, and **from_seconds()**. Duration_t has two new static constants: Duration_t.INFINITE and Duration_t.ZERO. Similarly, Time_t also has two new static constants: Time_t.INFINITE and Time_t.ZERO.

4 Performance and Resource Usage Improvements

4.1 Increased TCP Transport Performance and Scalability in Windows

This release increases the performance and scalability of the TCP transport on Windows systems by adding I/O Completion Ports (IOCP) support.

To configure the TCP transport to use IOCP, set the new property **socket_monitoring_kind** to WINDOWS_IOCP. For details, see Section 36.2.5, *TCP/TLS Transport Properties*, in the *Core Libraries and Utilities User's Manual*.

4.2 Memory Usage Reduction in rtiddsspy for Types with Large Maximum Size

In previous releases, *rtiddsspy* preallocated a set of buffers to receive incoming samples from *DataWriters*. The size of these buffers was equal to the maximum serialized size of the types associated with the incoming samples.

For types with a large maximum size, *rtiddsspy* could run out of memory even when the actual size of the incoming samples was small.

This release resolves this problem by preallocating the buffers to a small size and using dynamic memory allocation if the size of the buffers is not enough to hold the incoming samples.

5 XML-Related Features

5.1 Support for Environment Variables in XML Configuration Files Attributes

The previous release introduces the ability to refer to an environment variable within an XML tag. This release extends this functionality by also allowing environment variables within attributes, keeping the same format.

For example:

```
<element attr="Attribute is $(MY_ATTRIB)">
    <name>The name is $(MY_NAME)</name>
    <value>The value is $(MY_VALUE)</value>
</element>
```

Providing support for environment variables within attributes increases reusability for a wider set of schemas. For instance, the feature can be used in XML-Based Application Creation to parameterize the domain ID and avoid file duplication for each domain.

5.2 Ability to Specify NULL in XML for Fields in EntityNameQosPolicy

The new default value for the EntityNameQosPolicy's **name** field is NULL. To support this in an XML configuration, it is now possible to set both the **name** and **role_name** fields of the EntityNameQosPolicy to NULL with the attribute **xsi:nil**. For example:

```
<participant_name>
  <name xsi:nil="true"/>
  <role_name xsi:nil="true"/>
</participant_name>
```

5.3 Schema for XML Application Description Now Allows Colon in Property Name

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

6 Code Generation

6.1 New Code Generator

This release includes a separate and new implementation of the Code Generator; this new version significantly improves the performance and the original implementation and makes easier to customize the generated output.

The new code generator can be invoked using the script *rtiddsgen2*. It coexists with the original implementation, *rtiddsgen*.

For more information, see the *RTI Code Generator 2 Getting Started Guide*.

Note: Code Generator 2 is considered an Early Access Release.

6.2 New Option in rtiddsgen to Configure Name of Macro for Exporting Symbols when Building Windows DLL

This release introduces a new *rtiddsgen* command-line option, **-dllExportMacroSuffix**, that allows you to configure the suffix of the macro used to export type-plugin symbols when building a Windows DLL.

If you run *rtiddsgen* without this option, the macro is named **NDDS_USER_DLL_EXPORT**. With this option, the macro is named **NDDS_USER_DLL_EXPORT_<*Suffix*>**.

7 Transport-Related Features

7.1 New UDPv4 Transport Property to Allow Communication Across NAT-Enabled IP Routers

This release includes a new UDPv4 transport property, **dds.transport.UDPv4.builtin.public_address**, which allows you to configure the IP address that a *DomainParticipant* will announce to other *DomainParticipants*.

This property will typically be initialized with the public address of the IP NAT router that provides access to the WAN.

For additional information on this new property, see Table 15.2 in the *RTI Core Libraries and Utilities User's Manual*.

7.2 Transport Information Propagated in Participant Discovery Data

Starting with this release, the **message_size_max** of each installed transport in a *DomainParticipant* is propagated as part of the participant discovery information.

A *DomainParticipant* can access the transport information of a remote *DomainParticipant* by inspecting the field **transport_info** in the remote *DomainParticipant's* Built-in Topic Data. For more information, see Chapter 16, Built-in Topics, in the *RTI Core Libraries and Utilities Users Manual*.

Connext uses the transport information propagated via discovery to detect potential misconfigurations in a *Connext* distributed system. If two *DomainParticipants* that discovered each other have one common transport with different values for **message_size_max**, *Connext* will print an error message indicating that condition.

8 Content-Filtering Features

8.1 Ability to Change Expression in ContentFilteredTopic

In previous releases, the filter expression in a ContentFilteredTopic was immutable and you could only change the expression parameters. This limitation has been removed. Now you can change the filter expression by means of the DDS_ContentFilteredTopic's **set_expression()** operation.

8.2 Ability to Define Float Constants in Filter Expressions

Now you can specify a float constant in a filter expression by using the suffix **F** (or **f**). For example:

```
"a_float = 4.3F"
```

Typically, F is only required if a double has been assigned a float value in the code and then is tested against a constant in a SQL expression. In this case, it is important to use the F suffix appropriately. Consider the following case:

```
struct MyType { float a_float; double a_double; };
```

Assign the values in code (C):

```
a_float = 4.1;
a_double = 4.1F;
```

This expression works because **a_float** is a float, so **a_double** is demoted to a float:

```
"a_float = a_double"
```

However, this next expression will not work because both operands are treated as doubles, but **a_double** was assigned a float:

```
"a_double = 4.1"
```

Instead, the expression must be written as:

```
"a_double = 4.1F"
```

9 Other Features and Improvements

9.1 Prioritized Samples Supported in Java and C# APIs

The *Prioritized Samples* feature is now supported in the Java and C# APIs; previously it was only available in the C and C++ APIs.

This feature allows you to prioritize traffic that is in competition for transmission resources. For more information, see Section 6.6.4, *Prioritized Samples*, in the *RTI Core Libraries and Utilities User's Manual*.

9.2 Ability to Zero Out Padding in Messages on the Wire

This release adds a new field to the DDS_TypeSupportQosPolicy, **cdr_padding_kind**. It can be set so that all padding bytes are set to zero during the serialization of CDR streams that are sent on the wire. By default, the padding will not be set to zero, which is the behavior in all previous releases.

9.3 EntityNameQosPolicy Now Applies to Publishers and Subscribers

Publishers and *Subscribers* can be now identified with a name, represented by two new members **publisher_name** and **subscriber_name** in PublisherQos and SubscriberQos, respectively.

Unlike names for *DomainParticipants*, *DataWriters* and *DataReaders*, the names for *Publishers* and *Subscribers* names are not propagated as part of the discovery information (therefore RTI tools will not display their values).

9.4 New Default Value for DomainParticipant's resource_limits.participant_property_string_max_length

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

9.5 New Default Value for DomainParticipant's participant_name.name

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

10 New 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.

Experimental features are clearly documented as such in this document or in the *Release Notes* document of the component in which they are included, as well as in the component's *User's Manual*.

For more information on the concept of experimental features, see the *RTI Core Libraries and Utilities Release Notes*.

10.1 Experimental: Turbo Mode For DataWriter Performance

This release provides a new experimental feature known as *Turbo Mode*. This feature greatly improves the performance of *Connext DDS* in situations where the application is writing messages at high-rate. Turbo Mode intelligently uses message-aggregation (so called *Connext DDS* batching feature) to combine any small samples written by a *DataWriter* resulting in fewer protocol messages and a much reduced CPU load. The lower CPU load results on both higher throughput and lower latency.

Turbo Mode uses an intelligent algorithm that automatically adjusts the number of bytes in a batch at run time according to current system conditions, such as write speed (or write frequency) and sample size. This intelligence is what gives it the ability to increase throughput at high message rates and avoid negatively impacting message latency at low message rates.

To enable Turbo mode, set the new property **dds.data_writer.enable_turbo_mode**. Turbo mode is not enabled by default.

For details on using these features, see Section 6.3.18 in the *RTI Core Libraries and Utilities User's Manual*.

10.2 Experimental: Auto Throttling For DataWriter Performance

This release provides a new experimental feature known as *Auto Throttling*. You can configure a *DataWriter* to automatically adjust the writing rate and the send window size to provide the best latency/throughput tradeoff as system conditions change.

To enable Auto Throttling, set the new properties **dds.domain_participant.auto_throttle.enable** and **dds.data_writer.auto_throttle.enable** to true. Auto Throttling is not enabled by default and only applies to reliable communications.

For details on using this feature, see Section 10.4, *Auto Throttling for DataWriter Performance—Experimental Feature*, in the *RTI Core Libraries and Utilities User's Manual*.

10.3 New Features in RTI Prototyper

RTI Prototyper is an experimental tool to accelerate *Connext* application development and scenario testing. It gives you an easy way to try out realistic scenarios on your own computer systems and networks, and get immediate information on the expected performance, resource usage, and behavior on your system.

This release provides the following new features in *Prototyper*:

10.3.1 Prototyper Now Includes Lua Scripting Engine

RTI Prototyper now includes an embedded Lua scripting engine. This fast, lightweight, and extensible scripting language provides an easy and powerful way to define the behavior of your distributed application components.

Configuring *Prototyper* to use Lua Scripting is straightforward and allows you to define custom behavior for all the entities and data contained in your XML configuration file. This enables the use of Rapid Application Development (RAD) techniques for developing sophisticated application to process data on the fly.

See the *RTI Prototyper Getting Started Guide* for more information (in **<*Connext installation directory*>/ndds.<*version*>/doc/pdf**).

10.3.2 Prototyper Starts Right Away if Only One Configuration is Provided

If the XML profiles contain only one configuration, *RTI Prototyper* will no longer prompt the user to choose one. Instead it will start right away, using the configuration. If multiple configurations are available, the user will be asked to choose one.

10.3.3 Prototyper -domainID Option Overrides Domain ID Specified in XML Configuration

The domain ID specified in the XML configuration can be now overwritten by using *RTI Prototyper* 's new command-line option, **-domainId**.