RTI Connext

Core Libraries

Release Notes

Version 7.1.0

Contents

Chapter 1 Introduction

RTI® *Connext*® 7.1.0 is a feature release based on release 7.0.0. See the Connext Releases web page on the RTI website for more information.

This document includes the following:

For an overview of new features in 7.1.0, see RTI Connext Core Libraries What's New in 7.1.0.

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

- **Use the RTI Customer Portal** (https://support.rti.com) to download RTI software 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 Forum** (https://community.rti.com) provides a wealth of knowledge to help you use *Connext*, including:
  - Documentation, at https://community.rti.com/documentation
  - 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.
- Performance benchmark results for *Connext* are published online at http://www.rti.com/products/dds/benchmarks.html. Updated results for new releases are typically published within two months after general availability of that release.

Chapter 2 System Requirements

2.1 Introduction

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

In this context, a host is the computer on which you will be developing a *Connext* application. A target is the computer on which the completed application will run. A host installation provides the *RTI Code Generator* 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.

Supported platforms, for all products in the *Connext* suite are listed in these tables:

- Table 1 Supported Platforms for Compiler-Dependent Products on the next page.
  This table is for products that are compiled into your application.
- Table 2 Supported Platforms for Connext Tools on page 6
  This table is for products that are ready to use, no compilation required.

Early Access releases are intended to showcase the latest *Connext* features; they support a smaller subset of platforms in comparison to LTS releases. The upcoming LTS release shall support a larger number of platforms.

Subsequent Early Access and LTS releases may not support all of the platforms supported in this release, or may support different versions of platforms supported in this release.

See the *RTI Connext Core Libraries Platform Notes* for more information on each platform.

2.2 Supported Platforms

Table 1 Supported Platforms for Compiler-Dependent Products

Platforms					**Connext Professional**							
							Secured Transport		Infrastructure Services			
OS	OS Version	CPU	Toolchain	RTI Architecture	Core Libraries	LBED [10]	TLS Support for OpenSSL 1.1.1	TLS Support for OpenSSL 3.0	Persistence Service	Routing Service	Recording Service	Web Integration Service
Windows	Windows 10 [3] Windows 11 Server 2016	x64	VS 2017, 2019, 2022	x64Win64VS2017	●	●	●	●	●	●	●	●
	Windows 10 Windows Server 2012 R2, 2016	x64	VS 2015	x64Win64VS2015	●	●	●	●	●	●	●	●
macOS	macOS 11, 12 (host and target) [9]	x64	clang 12.0, 13.0	x64Darwin20clang12.0	●		●	●	●	●	●	●
	macOS 11, 12 (target only) [11]	Arm v8	clang 12.0, 13.0	arm64Darwin20clang12.0	●		●	●	●	●	●	●
Linux	Red Hat Enterprise Linux 8, 9 Ubuntu 18.04 LTS, 20.04 LTS, 22.04 LTS	x64	gcc 7.3.0	x64Linux4gcc7.3.0	●	●	●	●	●	●	●	●
	Red Hat Enterprise Linux 7, 7.3, 7.5, 7.6 CentOS 7.0	x64	gcc 4.8.2	x64Linux3gcc4.8.2	●	●	●	●	●	●	●	●
	Ubuntu 18.04 LTS, 22.04 LTS	Arm v8 [2]	gcc 7.3.0	armv8Linux4gcc7.3.0	●		●	●		●	●	●
QNX	QNX Neutrino 7.1	Arm v8[2]	qcc 8.3.0	armv8QNX7.1qcc_gpp8.3.0	●		●	●		●	●	

Table 1 (Continued) Supported Platforms for Compiler-Dependent Products

Platforms					Connext Secure				Connext Anywhere		Add-ons	
OS	OS Version	CPU	Toolchain	RTI Architecture	Security Plugins with OpenSSL 1.1.1 [1]	Security Plugins with OpenSSL 3.0 [12]	Security Plugins for wolfSSL 5.5 [8]	Security Plugins SDK [1, 12]	Cloud Discovery Service	Real-Time WAN Transport	Limited Bandwidth Plugins	Observability Framework
Windows	Windows 10 [3] Windows 11 Server 2016	x64	VS 2017, 2019, 2022	x64Win64VS2017	●	●		●	●	●	●	●
	Windows 10 Windows Server 2012 R2, 2016	x64	VS 2015	x64Win64VS2015	●	●		●	●	●	●	●
macOS	macOS 11, 12 (host and target) [9]	x64	clang 12.0, 13.0	x64Darwin20clang12.0	●	●		●	●	●		●
	macOS 11, 12 (target only) [11]	Arm v8	clang 12.0, 13.0	arm64Darwin20clang12.0	●	●			●	●		●
Linux	Red Hat Enterprise Linux 8, 9 Ubuntu 18.04 LTS, 20.04 LTS, 22.04 LTS	x64	gcc 7.3.0	x64Linux4gcc7.3.0	●	●		●	●	●	●	●
	Red Hat Enterprise Linux 7, 7.3, 7.5, 7.6 CentOS 7.0	x64	gcc 4.8.2	x64Linux3gcc4.8.2	●	●		●	●	●	●	●
	Ubuntu 18.04 LTS, 22.04 LTS	Arm v8 [2]	gcc 7.3.0	armv8Linux4gcc7.3.0	●	●				●		●
QNX	QNX Neutrino 7.1	Arm v8[2]	qcc 8.3.0	armv8QNX7.1qcc_ gpp8.3.0	●	●	●			●		●

Table 2 Supported Platforms for Connext Tools

Platforms			Tools				
OS	OS Version	CPU	Shapes Demo	Launcher	Monitor	Admin Console	System Designer
Windows	Windows 10 [3] Windows 11 Windows Server 2016 [3]	x64	●	●	●	●	● [7]
	Windows 10 Windows Server 2012 R2, 2016	x64	●	●	●	●	● [7]
macOS	macOS 11, 12 (host and target) [9]	x64	●	●	●	●	● [6]
	macOS 11, 12 (target only) [11]	Arm v8					
Linux	Red Hat Enterprise Linux 8, 9 Ubuntu 18.04 LTS, 20.04 LTS, 22.04 LTS	x64	●	●	●	●	● [5]
	Red Hat Enterprise Linux 7, 7.3, 7.5, 7.6 CentOS 7.0	x64	●	●	●	●	● [5]
	Ubuntu 18.04, 22.04 LTS	Arm v8 [2]					
QNX	QNX Neutrino 7.1	Arm v8 [2]					

● = Supported

1 Tested with OpenSSL 1.1.1t
2 These libraries require a hardware FPU in the processor and are compatible with systems with hard-float libc
3 Per Microsoft, this should be compatible with Windows 10 IoT Enterprise with Windows native application
5 Tested on Ubuntu 18.04 LTS only, with Chrome 77, Firefox 69
6 Tested on macOS 10.14 only, with Chrome 77, Firefox 69, and Safari 12
7 Tested on Windows 10 only, with Chrome 77 and Firefox 69
8 Tested with wolfSSL 5.5.1
10 LBED = Limited Bandwidth Endpoint Discovery Plugin
11 Requires Rosetta® 2 during installation, not required at runtime
12 Tested with OpenSSL 3.0.8

2.3 Requirements when Using Microsoft Visual Studio

Note: Debug versions of applications and the various Visual C++ DLLs are not redistributable. Therefore, if you want to run debug versions, you must have the compiler installed.

When Using Visual Studio 2015 — Update 3 Redistributable Package Requirement

You must have the Visual C++ Redistributable for Visual Studio 2015 Update 3 installed on the machine where you are running an application linked with dynamic libraries. This includes C/C++ dynamically linked and all Java applications.

You can download the Visual C++ Redistributable for Visual Studio 2015 Update 3 from this Microsoft website: https://www.microsoft.com/en-us/download/details.aspx?id=53840.

When Using Visual Studio 2017 — Redistributable Package Requirement

You must have the Visual C++ Redistributable for Visual Studio 2017 installed on the machine where you are running an application linked with dynamic libraries. This includes C/C++ dynamically linked and all Java applications.

You can download the Visual C++ Redistributable for Visual Studio 2017 from this Microsoft website: https://visualstudio.microsoft.com/vs/older-downloads/. Then look in this section: "Redistributables and Build Tools" for Microsoft Visual C++ Redistributable for Visual Studio 2017".

When Using Visual Studio 2019 — Redistributable Package Requirement

You must have the Visual C++ Redistributable for Visual Studio 2019 installed on the machine where you are running an application linked with dynamic libraries. This includes C/C++ dynamically linked and all Java applications.

You can download the Visual C++ Redistributable for Visual Studio 2019 from this Microsoft website: https://www.visualstudio.com/downloads/. Then look in this section: "Other Tools and Frameworks" for Microsoft Visual C++ Redistributable for Visual Studio 2019".

When Using Visual Studio 2022 — Redistributable Package Requirement

You must have the Visual C++ Redistributable for Visual Studio 2022 installed on the machine where you are running an application linked with dynamic libraries. This includes C/C++ dynamically linked and all Java applications.

You can download the Visual C++ Redistributable for Visual Studio 2019 from this Microsoft website: https://www.visualstudio.com/downloads/. Then look in this section: "Other Tools, Frameworks, and Redistributables" for Microsoft Visual C++ Redistributable for Visual Studio 2022".

2.4 Disk and Memory Usage

Disk usage for a typical host-only installation is approximately 802 MB on Linux systems and 821 MB on Windows systems. Each additional architecture (host or target) requires an additional 498 MB on Linux systems and 609 MB on Windows systems.

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

Chapter 3 Compatibility

Below is basic compatibility information for this release.

> **Note:** For backward-compatibility information between this and previous releases, see the *Migration Guide* on the RTI Community Portal (https://community.rti.com/documentation).

3.1 Wire Protocol Compatibility

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

RTPS 1.0 was introduced in 2001. The currently supported version is OMG Real-Time Publish-Subscribe (RTPS) specification, version 2.5, although some features are not supported. Unsupported features currently are FilteredCountFlag in GAP Submessage, HeartbeatFrag Submessage, Checksum, and ALIVE_FILTERED instance state. RTI plans to maintain interoperability between middleware versions based on RTPS 2.1. For more details, see Table 3.1 RTPS Versions.

Table 3.1 RTPS Versions shows RTPS versions supported for each *Connext* release. In general, RTPS 2.1 and higher versions are interoperable, unless noted otherwise. RTPS 2.0 and RTPS 1.2 are incompatible with current (4.2e and later) versions of *Connext*.

Although RTPS 2.1 and higher versions are generally interoperable, there may be specific wire protocol interoperability issues between *Connext* releases. These issues are documented in the "Wire Protocol" section for your release, in the *Migration Guide* on the RTI Community Portal (https://community.rti.com/documentation). Wire protocol issues between 5.3.1 and previous releases are documented in the *RTI Connext Core Libraries Release Notes* for release 5.3.1.

Table 3.1 RTPS Versions

Connext Release	RTPS Standard Version (a)	RTPS Protocol Version on the Wire (b)
Connext 7.1.0 and above	2.5 (partial support)	2.5
Connext 6 and 7.0.0	2.3 (partial support)	2.3
Connext DDS 5.2 and 5.3	2.2 (partial support)	2.1
Connext DDS 4.5f - 5.1	2.1	2.1
Data Distribution Service 4.2e - 4.5e	2.1	2.1
Data Distribution Service 4.2c	2.0	2.0
Data Distribution Service 4.2b and lower	1.2	1.2

(a) Version number of the RTPS standards document, OMG Real-Time Publish-Subscribe (RTPS) specification, version 2.5.

(b) RTPS wire protocol version number that *Connext* announces in messages it puts on the wire.

3.2 Code and Configuration Compatibility

The *Connext* core uses an API that is an extension of the OMG Data Distribution Service (DDS) standard API, version 1.4. RTI strives to maintain API compatibility between versions, but will conform to changes in the OMG DDS standard.

The *Connext* core 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 the *Migration Guide* on the RTI Community Portal (https://community.rti.com/documentation). The *Migration Guide* also indicates whether and how to regenerate code.

3.3 Extensible Types Compatibility

This release of *Connext* includes partial support for the OMG 'Extensible and Dynamic Topic Types for DDS' specification, version 1.3 (DDS-XTypes) from the Object Management Group (OMG). This support allows systems to define data types in a more flexible way, and to evolve data types over time without giving up portability, interoperability, or the expressiveness of the DDS type system.

For information related to compatibility issues associated with the Extensible Types support, see the *Migration Guide* on the RTI Community Portal (https://community.rti.com/documentation). See also the *RTI Connext Core Libraries Extensible Types Guide* for a full list of the supported and unsupported extensible types features.

Chapter 4 What's Fixed in 7.1.0

This section describes bugs fixed in *Connext* 7.1.0. These fixes have been made since 7.0.0 was released.

4.1 Fixes Related to Discovery

4.1.1 Potential memory leak when creation of any of the built-in discovery plugins failed

The first time a *DomainParticipant* is created in an application, some memory is allocated globally for each of the built-in discovery plugins (SPDP and SEDP) enabled for that *DomainParticipant*. This global memory is released when finalizing the *DomainParticipantFactory* instance.

However, if there was a failure in the creation of any of the builtin discovery plugins during the *DomainParticipant* creation, the *DomainParticipantFactory* was not notified properly that this global memory was allocated. Therefore, finalizing the *DomainParticipantFactory* instance did not release the memory, causing a leak.

This problem is fixed. Finalizing the *DomainParticipantFactory* instance always releases the memory if it was previously allocated, regardless of whether or not a failure occurred.

[RTI Issue ID CORE-12882]

4.1.2 Unbounded memory growth when using domain tags or DomainParticipant partitions

Whenever a *DomainParticipant* discovered another *DomainParticipant* that it did not match with, either due to a mismatched domain tag or participant partition, some state was kept that was never removed if the *DomainParticipant* never received an announcement from that same mismatched participant indicating that it had been shut down. This led to unbounded memory growth, which could become an issue in systems where *DomainParticipants* with various different domain tags or partitions were coming and going.

[RTI Issue ID CORE-12973]

4.1.3 Error deleting remote endpoints with specific GUID prefixes using debug libraries

An error occurred when using debug libraries in the unlikely case that a *DomainParticipant* had a zero value as the hostId, appId, or instanceId. This problem has been fixed.

[RTI Issue ID CORE-13261]

4.1.4 Most up-to-date participant configuration may not have been received by other participants and may have led to discovery not completing

It was possible that a configuration change in *DomainParticipant* 'A' may not have been received by *DomainParticipant* 'B' if the change occurred while the two participants were discovering each other. Examples of configuration changes are a change in the PROPERTY QoS policy or an IP mobility event in which *DomainParticipant* 'A' changes one of its IP addresses.

Not having the most recent configuration may have led to discovery not happening if the change was due to an IP mobility event.

The problem only occurred when discovery used multiple transports (e.g, SHMEM and UDPv4). This problem has been fixed.

[RTI Issue ID CORE-13359]

4.1.5 Participant failed to assert remote participant if usability of shared memory transport changed

In 7.0.0, a DomainParticipant failed to assert a remote DomainParticipant if the usability of the shared memory transport changed, resulting in the following log message:

```
ERROR [0x010114FE,0x12488672,0x8EE3B6BC:0x000100C7{Entity=DR,MessageKind=DATA}|RECEIVE FROM
0x00000000,0x00000000,0x00000000:0x000100C2|:0x000001C1{Domain=0}|ASSERT REMOTE
DP|LC:DISC]PRESParticipant_assertConfiguredRemoteParticipant:ASSERT FAILURE | compare
immutable remote participant 0x01017851,0x3B428DDD,0x514330AA config RW
ERROR [0x010114FE,0x12488672,0x8EE3B6BC:0x000100C7{Entity=DR,MessageKind=DATA}|RECEIVE FROM
0x00000000,0x00000000,0x00000000:0x000100C2|LC:DISC]DISCParticipantDiscoveryPlugin_
assertRemoteParticipantConfig:!assert remote participant:
0x01017851,0x3B428DDD,0x514330AA,0x000001C1
ERROR [0x010114FE,0x12488672,0x8EE3B6BC:0x000100C7{Entity=DR,MessageKind=DATA}|RECEIVE FROM
0x00000000,0x00000000,0x00000000:0x000100C2|LC:DISC]DISCParticipantDiscoveryPlugin_
assertRemoteParticipantFull:ASSERT FAILURE | remote participant
0x01017851,0x3B428DDD,0x514330AA config information
ERROR [0x010114FE,0x12488672,0x8EE3B6BC:0x000100C7{Entity=DR,MessageKind=DATA}|RECEIVE FROM
0x00000000,0x00000000,0x00000000:0x000100C2|LC:DISC]PRESParticipantAnnouncementChannelReader
ListenerSpdp_onDataAvailable:!assert remote participant
```

You may have run into this issue if a shared memory segment was deleted during runtime and a DomainParticipant updated its configuration information. A change in the shared memory usability will no longer cause this failure.

[RTI Issue ID CORE-13360]

4.1.6 Unexpected warning during discovery when multicast disabled

Connext logged a warning during the discovery process when multicast was disabled. The message warned about unreachable multicast locators. The message was unexpected and has been removed.

[RTI Issue ID CORE-13403]

4.1.7 Unexpected, invalid locator propagated within builtin topics

A *DataReader* could unexpectedly propagate an invalid locator to a *DataWriter* for certain builtin topics. The issue did not affect functionality, since the locator was discarded on the *DataWriter* side. The bug that sent the invalid locator has been fixed.

[RTI Issue ID CORE-13416]

4.2 Fixes Related to Serialization and Deserialization

4.2.1 Unexpected union value when receiving a discriminator that does not select any union member on DataReader's type

When the property **dds.sample_assignability.accept_unknown_union_discriminator** was set to 1, previous *Connext* releases were not always compliant with the latest OMG 'Extensible and Dynamic Topic Types for DDS' specification, version 1.3 when a *DataWriter* publishes a union sample with a discriminator value that selects a union member, and a *DataReader* subscribes to a union type that does not have a union member for the discriminator published by the *DataWriter*.

For example:

```
/* Publisher */
union MyUnion switch(int32) {
    case 0:
        int32 m1;
    case 1:
        int16 m2;
    case 2:
        double m3;
};

/* Subscriber */
union MyUnion switch(int32) {
    case 0:
        int32 m1;
    case 1:
        int16 m2;
};
```

In this example, if the *DataWriter* published a sample with a discriminator value set to 2 selecting m3, the *DataReader* received a sample where the discriminator is set to 0 and m1 is set to 0, the default value of the union. According to the OMG 'Extensible and Dynamic Topic Types for DDS' specification, version 1.3, the *DataReader* should preserve the discriminator value received from the *DataWriter* even if this discriminator value does not select any member in the *DataReader's* union.

This problem only occurred when one of these conditions was true:

- The unions are mutable regardless of the data encapsulation (XCDR1 or XCDR2).
- The unions are appendable, and the encapsulation is XCDR2.

Note if the union discriminator did not select any member on the *DataWriter's* type, such as 3 in the above example, the *DataReader* received the expected discriminator 3.

This release accepts a new value for the **dds.sample_assignability.accept_unknown_union_discriminator** property:

- 0 (existing value and default value): Received samples containing a union discriminator value that selects a union member on the *DataWriter* but not on the *DataReader* are dropped.
- 1 (existing value) : Received samples containing a union discriminator value that selects a union member on the *DataWriter* but not on the *DataReader* are set to the default union value.
- 2 (new value): Received samples containing a union discriminator value that selects a union member on the *DataWriter* but not on the *DataReader* preserve the discriminator value.

Received samples containing a union discriminator value that does not select a union member on the *DataWriter* always preserve the discriminator value on the *DataReader* with **dds.sample_assignability.accept_unknown_union_discriminator** set to 1 or 2, unless the union discriminator value is an enumerator which is not valid on the *DataReader's* type. In this case, the union is set to its default value.

To be compliant with the OMG 'Extensible and Dynamic Topic Types for DDS' specification, version 1.3, set the value to 2.

[RTI Issue ID CORE-13058]

4.2.2 Serialization of samples failed or produced a segmentation fault for types with max serialized size larger than 2GB

A *DataWriter* may have failed to send a sample due to serialization errors when the sample's type had a max serialized size with a value larger than 2GB.

For example:

```
@nested
    struct MyNestedStruct2 {
        sequence<octet, 1500000000> m1;
    };

    @nested
    struct MyNestedStruct {
        sequence<octet, 1000000000> m1;
        MyNestedStruct2 m2;
    };

    struct MyStruct {
        MyNestedStruct m1;
    };
```

In this example, the serialize operation failed with an error like this:

```
[0x0101C50B,0x0D4E0B41,0xBBFA04AC:0x80000003{E=DW,T=Example MyStruct,C=MyStruct,D=56}|WRITE]
PRESWriterHistoryDriver_serializeSample:serialize sample error in topic 'Example MyStruct'
with type 'MyStruct' and encapsulationId 1
```

For 32-bit platforms, the application may have produced a segmentation fault instead of failing to serialize.

This problem has been fixed.

[RTI Issue ID CORE-12687]

4.2.3 Potential sample corruption when deserializing a malformed RTPS message

A sample could be corrupted/incomplete with no error logged in the case of a deserialization failure in the transport info parameter of the RTPS message. This problem has been fixed.

[RTI Issue ID CORE-13366]

4.2.4 Unbounded memory growth when deserializing a malformed RTPS message

Potential unbounded memory growth occurred while parsing a malicious RTPS message. This problem has been fixed.

[RTI Issue ID CORE-13397]

4.3 Fixes Related to Debuggability

4.3.1 Hang/crash when invoking a DataReader/DataWriter discovery snapshot within a callback function

A hang or even a crash occurred when trying to get a discovery snapshot from a *DataReader* or *DataWriter* within a callback. RTI strongly recommends avoiding calling discovery snapshot APIs in callback functions in release 7.0.0. This issue has been fixed in 7.1.0.

[RTI Issue ID CORE-12959]

4.3.2 Memory leak if network capture initialization failed

Failure to initialize network capture for a *DomainParticipant* may have caused a memory leak of 746 kB. The leak only happened (upon *DomainParticipant* creation) if the initialization failed when creating the status mutex for a manager:

```
!create status mutex for the network capture manager
```

This issue is now fixed. A failure creating the status mutex for a manager does not leak memory anymore.

[RTI Issue ID CORE-13018]

4.3.3 Unexpected log messages at warning verbosity

You may have seen the following unexpected log messages at the warning verbosity level:

```
!get xxx remoteWriter
!get xxx remoteReader
!goto WR xxx remote reader
!goto WR xxx remote writer
```

These warnings did not signal any unexpected scenario, and they have been removed.

[RTI Issue ID CORE-13434]

4.3.4 Unexpected fatal error when number of instances reached the limit

In 7.0.0, an unexpected fatal error could be logged when the following occurred:

- A *DataWriter* is configured to use durable writer history.
- The number of instances reached the **max_instances** limit set in the *DataWriter's* RESOURCE_LIMITS QoS.
- *Connext* could not find an instance to delete (such as an unregistered one), to replace with the new instance. So the new instance could not be added.

This log message is expected, but it is not a fatal error, so its verbosity has been updated to WARNING, as follows:

```
WriterHistoryOdbcPlugin_createResources:FIND FAILURE | Instance for replacement
WriterHistoryOdbcPlugin_addInstance:OUT OF RESOURCES | Exceeded the number of instances.
Current registered instances (128), maximum number of instances (128)(writer_qos.resource_
limits.max_instances)
```

[RTI Issue ID CORE-13496]

4.4 Fixes Related to Transports

4.4.1 Possible data loss after a Connext application lost its multicast interfaces or gained its first multicast interface

The IP mobility feature detects when the interfaces of an application change, then propagates these changes. If an IP mobility event causes either the loss of the last interface that supported multicast or the gain of the first interface that supports multicast, the way other applications communicate with the application that experienced the IP mobility event changes.

Previously, that transition did not happen properly and may have led to data losses. This problem has been fixed. Now, communication is not affected by these interface changes.

[RTI Issue ID CORE-12609]

4.4.2 DomainParticipant with non-default metatraffic_transport_priority QoS did not complete discovery

A *DomainParticipant* that had a non-default **metatraffic_transport_priority** in the DISCOVERY QoS Policy was not able to complete endpoint discovery due to a unicast metatraffic channel that was not created correctly. (The channel is used by the participant to send Data(R) and Data(W).)

This issue was introduced in 6.1.0. This issue has been resolved.

[RTI Issue ID CORE-12739]

4.4.3 dds.transport.minimum_compatibility_version property did not properly adjust locator format

Connext 5.3.0 introduced a new shared memory locator format. *DomainParticipants* in *Connext* 5.3.0 (and above) use the new locator format by default. To allow interoperability with *Connext* versions before 5.3.0, you must indicate to *DomainParticipants* to use the old locator format.

There are two properties for telling a *DomainParticipant* to use the old locator format: **dds.transport.use_530_shmem_locator_matching** (undocumented and deprecated) and **dds.transport.minimum_compatibility_version**. The latter is a newer property that combines several other properties. Its purpose is to set the transport to be compatible with the specified version in a simplified manner.

The problem with the newer property, **dds.transport.minimum_compatibility_version**, was that it did not adjust the locator format depending on the *Connext* version. The workaround was to use the **dds.transport.use_530_shmem_locator_matching** property instead. This issue has been fixed. You can now use **dds.transport.minimum_compatibility_version** without issue.

[RTI Issue ID CORE-12789]

4.4.4 TCP Transport did not run with Windows debug libraries when socket_ monitoring_kind was set to IOCP

An internal error prevented the TCP transport from running on Windows with debug libraries when the **socket_monitoring_kind** was set to the recommended value of NDDS_TRANSPORT_TCPV4_ SOCKET_MONITORING_KIND_WINDOWS_IOCP. The error has been corrected.

[RTI Issue ID COREPLG-654]

4.5 Fixes Related to Reliability Protocol and Wire Representation

4.5.1 Samples not delivered to Required Subscription DataReaders when DataWriter used durable writer history and DataReaders disabled positive ACKs

A sample may not have been delivered to a Required Subscription *DataReader* if the *DataWriter* was using durable writer history and there were matching *DataReaders* configured with **reader_qos.-protocol.disable_positive_acks**. This behavior violated the required subscription contract. This problem has been resolved.

[RTI Issue ID CORE-12825]

4.5.2 DataReader may not have received samples that were sent as gapped samples to another DataReader over multicast

A *DataReader* may not have received samples that were sent as gapped samples to another *DataReader* over multicast. A GAP tells a *DataReader* that it should not expect to receive the samples that are listed in the GAP message. In some cases, when a *DataWriter* was responding to a *DataReader's* NACK message, the response contained a GAP which identified samples that should not have been gapped for any other *DataReader* aside from the *DataReader* whose NACK was being responded to. This was a problem if the NACK response was sent over multicast and was received by other *DataReaders*, because those *DataReaders* would incorrectly assume those gapped samples were irrelevant and would never receive them.

This issue has been resolved.

[RTI Issue ID CORE-13104]

4.5.3 DDS fragmentation may have led to more fragments than expected for a sample

In 7.0.0, you may have noticed that when using middleware-level fragmentation and a flow controller where **bytes_per_token** is set to a value smaller than the minimum transport **message_size_max** across all installed transports, the number of sample fragments generated for a sample may have been bigger than expected. Although this was not a functional issue, it may have led to performance degradation.

This problem has been fixed.

[RTI Issue ID CORE-13190]

4.5.4 Unexpected precondition error with debug libraries on a reliable DataWriter while sending a GAP

In the 6.1.2 and 7.0.0 releases, you may have seen the following precondition error while using the *Connext* debug libraries.

```
DL Debug: :     Backtrace:
141: DL Debug: :    #4  COMMENDSrWriterService_sendGapToRR /rti/jenkins/workspace/connextdds_
ci_fastbuild-debug_develop/commend.1.0/srcC/srw/SrWriterService.c:4096 (discriminator 9)
[0x5B101E]
141: DL Debug: :    #5  COMMENDSrWriterService_onSendDataEvent
/rti/jenkins/workspace/connextdds_ci_fastbuild-debug_
develop/commend.1.0/srcC/srw/SrWriterService.c:6570 [0x5BACF6]
141: DL Debug: :    #6  RTIEventActiveGeneratorThread_loop /rti/jenkins/workspace/connextdds_
ci_fastbuild-debug_develop/event.1.0/srcC/activeGenerator/ActiveGenerator.c:307 [0x28E2FC]
141: DL Debug: :    #7  RTIOsapiThreadFactory_onSpawned /rti/jenkins/workspace/connextdds_ci_
fastbuild-debug_develop/osapi.1.0/srcC/threadFactory/ThreadFactory.c:208 [0x1F3A42]
141: DL Debug: :    #8  RTIOsapiThreadFactory_onSpawned /rti/jenkins/workspace/connextdds_ci_
fastbuild-debug_develop/osapi.1.0/srcC/threadFactory/ThreadFactory.c:208 [0x1F3A42]
141: DL Debug: :    #9  RTIOsapiThreadChild_onSpawned /rti/jenkins/workspace/connextdds_ci_
fastbuild-debug_develop/osapi.1.0/srcC/thread/Thread.c:1941 [0x1EDB64]
141: DL Debug: :    #10 start_thread /build/glibc-CVJwZb/glibc-2.27/nptl/pthread_create.c:463
[0x76DB]
141: DL Debug: :    #11 clone /build/glibc-CVJwZb/glibc-
2.27/misc/../sysdeps/unix/sysv/linux/x86_64/clone.S:97 [0x12161F]
141: DL Fatal: : FATAL rCoRTInk####Evt [0x01014F91,0x39810444,0x4EC68AEA:0x000004C2|RECEIVE
FROM remote DR (GUID: 0x01015FBD,0x5892DC7E,0x9DB082D4:0x000004C7).
141: ] Mx00:/rti/jenkins/workspace/connextdds_ci_fastbuild-debug_
develop/commend.1.0/srcC/srw/SrWriterService.c:4099:RTI0x200003b:!precondition: "
((((gapStartSn)->high) > (((&(gapBitmap)->_lead))->high)) ? 1 : ((((gapStartSn)->high) <
(((&(gapBitmap)->_lead))->high)) ? -1 : ((((gapStartSn)->low) > (((&(gapBitmap)->_lead))-
>low)) ? 1 : ((((gapStartSn)->low) < (((&(gapBitmap)->_lead))->low)) ? -1 : 0)))) >= 0"
141: DL Error: : ERROR [0x01014F91,0x39810444,0x4EC68AEA:0x000004C2|RECEIVE FROM remote DR
(GUID: 0x01015FBD,0x5892DC7E,0x9DB082D4:0x000004C7).
141: ] COMMENDSrWriterService_onSendDataEvent:!send GAP
```

This error was generated by a reliable *DataWriter* sending a GAP to a reliable *DataReader*. After the error was printed, the *DataReader* may have stopped receiving data from the *DataWriter*, leading to a non-recoverable situation. This problem did not occur with release libraries. This problem has been fixed.

[RTI Issue ID CORE-13462]

4.6 Fixes Related to Content Filters and Query Conditions

4.6.1 Unexpected "RTIXCdrSampleInterpreter_initializeSampleWInstruction" error log messages when using QueryConditions, ContentFilteredTopics, TopicQueries, or Multi-Channel

In releases 6.0.x and 6.1.x, a *Connext* application using QueryConditions, ContentFilteredTopics, TopicQueries, or Multi-Channel may have logged an error message like the following when applying filtering to some samples:

```
RTIXCdrSampleInterpreter_initializeSampleWInstruction: <Type>:<Field Name> initialize error
```

A potential workaround was to set the property **dds.content_filter.sql.deserialized_sample.min_buffer_size** to -1 in the participant_qos.**property** QoS Policy. However, this may have led to a higher memory utilization.

This problem has been resolved.

[RTI Issue ID CORE-13328]

4.7 Fixes Related to Dynamic Data

4.7.1 DynamicData DataWriters incorrectly serialized optional empty sequences as null

In previous 6.0.0 releases and above, a DynamicData *DataWriter* incorrectly serialized an optional empty sequence as null. When a *DataReader* received the sample, it deserialized the wrong value.

For example, assume the following type:

```
struct AuditLogEntry {
    long long Nanoseconds;
    @optional sequence<long long, 100> Details;
};
```

If the publishing application set Details to an empty sequence with zero elements, the serialized value was incorrectly set to null. When a *DataReader* received the sample, it incorrectly set Details to null instead of the empty sequence with zero elements.

This problem has been fixed.

[RTI Issue ID CORE-12866]

4.8 Fixes Related to APIs

4.8.1 DynamicData method to get member type missing in Modern C++ and C# APIs

The method to retrieve a member type from a DynamicData object was not provided in the Modern C++ and C# APIs. The following methods have now been added:

- C++: DynamicData::member_type(const std::string& name) and member_type(uint32_t id)
- C#: DynamicData.GetMemberType(string name) and GetMemberType(int id)

[RTI Issue ID CORE-13371]

4.8.2 Fixes Related to Modern C++ API

4.8.2.1 banish and subject_name APIs were unresolved in Modern C++ Windows dynamic libraries

The Modern C++ APIs **banish_ignored_participants**, **discovered_participant_subject_name**, and **discovered_participants_from_subject_name** were unresolved symbols in the nddscpp2 Windows dynamic libraries. If you attempted to use them, you would get LNK2019 unresolved external symbol errors. This problem has been fixed.

[RTI Issue ID CORE-13053]

4.8.2.2 Unnecessary small memory allocation in some operations, including read/take

Every call to a *DataReader* read/take operation caused an unnecessary small memory allocation that was immediately released. More generally, initializing a reference type to **dds::core::null** caused the same allocation. For example:

```
DomainParticipant p = dds::core::null;
```

This unnecessary allocation has been removed. Constructing a reference type to **dds::core::null** no longer allocates memory.

[RTI Issue ID CORE-13262]

4.8.2.3 close() operation of a ContentFilteredTopic created from XML didn't work

The **close()** operation of a ContentFilteredTopic created from XML didn't actually close it. However, when its *DomainParticipant* was closed or destroyed, the ContentFilteredTopic was correctly closed. This problem has been resolved.

[RTI Issue ID CORE-13367]

4.8.3 Fixes Related to C# API

4.8.3.1 Windows library dependency missing from .NET API NuGet packages

In release 7.0.0, Windows machines that did not have the Visual Studio redistributable may not have been able to run DDS .NET applications out of the box. This dependency is now managed internally and no longer required by the user.

[RTI Issue ID CORE-13120]

4.8.3.2 Exception when disposing a DomainParticipant or when entities were not properly disposed

In previous releases of the .NET API, an exception may have occurred when disposing a *DomainParticipant* or whenever unused entities that had not been properly disposed were garbage-collected.

[RTI Issue ID CORE-13231]

4.8.4 Fixes Related to Java API

4.8.4.1 Java API leaked some objects in certain DomainParticipantFactory operations

The Java API created and pinned a number of objects as a result of calling most methods in the DomainParticipantFactory, including the creation of *DomainParticipants*. While these objects did not consume significant amounts of memory, certain JVMs could have exhausted the maximum number of allowed global references, causing applications to fail. This problem has been resolved.

[RTI Issue ID CORE-12838]

4.8.4.2 get_typecode method of a DomainParticipant in Java API failed when the type contained a wstring element

In the Java API, calling the **get_typecode** method on a *DomainParticipant* for a registered type that contained a wstring element failed with the following exception:

```
Exception in thread "main" com.rti.dds.infrastructure.BAD_TYPECODE: Error creating type code
at com.rti.dds.typecode.TypeCodeFactory.create_tc_from_native(TypeCodeFactory.java:984)
at com.rti.dds.domain.DomainParticipantImpl.get_typecode(DomainParticipantImpl.java:2027)
```

The exception was caused by a problem in the way the *Connext* Java API interfaced with its internal C implementation. This problem has been resolved.

[RTI Issue ID CORE-13302]

4.8.5 Fixes Related to Python API

4.8.5.1 DynamicData accessor for an enum member in a base type failed (Python API)

Given a DynamicData for a struct type (my_struct) with a base type containing an enum member (my_enum), the following code failed:

```
sample = dds.DynamicData(my_struct)
print(sample["my_enum"]) # error: member my_enum doesn't exist
```

This problem has been resolved.

[RTI Issue ID PY-30]

4.8.5.2 Possible incorrect default values when receiving extensible data

Given the following situation:

- An application uses a **dds.DataReader** for an extensible IDL type "T1" containing a non-optional primitive member "a".
- The reader receives data for a different-but-compatible type "T2" that doesn't define "a".

The reader is expected to return a data sample where "a" is set to its default value (normally 0). However, in some situations the data sample may have contained an unexpected value for "a". This problem has been resolved.

[RTI Issue ID PY-77]

4.8.5.3 Some APIs where missing, incorrectly named, or have been deleted

4.8.5.3.1 Removed types, methods, and fields:

- TopicInstance and all related operations in the *DataReader* and *DataWriter* have been removed.
- The static properties **dds.WriterDataLifecycle.auto_dispose_unregistered_instances** and **dds.WriterdataLifecycle.manually_dispose_unregistered_instances** have been removed due to being too similar to the non-static properties.
- The *DataReader* operations **read_next** and **take_next** have been removed.

4.8.5.3.2 Renamed types, methods and fields:

- **dds.ReaderDataLifecycle.autopurge_unregistered_instances_delay** was incorrectly named and has been renamed to **autopurge_nowriter_samples_delay**; **autopurge_nowriter_instances_delay** was missing and has been added.

- **dds.Filter.sql_filter_name** has been renamed to **dds.Filter.SQL_FILTER_NAME**; **dds.Filter.stringmatch_filter_name** has been renamed to **dds.Filter.STRINGMATCH_FILTER_NAME**. The same constants have been renamed in **dds.MultiChannel**.
- **dds.DataWriterResourceLimitsInstaceReplacementKind** was misspelled and has been renamed to **dds.DataWriterResourceLimitsInstanceReplacementKind**.
- **dds.TransportMulticast.settings** has been renamed to **dds.TransportMulticast.value**; **dds.TransportMulticastMapping.settings** has been renamed to **dds.TransportMulticastMapping.value**; **dds.TransportSelection.enabled_transports** has been renamed to **dds.TransportSelection.value**; **dds.TransportUnicast.settings** has been renamed to **dds.TransportUnicast.value**.

4.8.5.3.3 Newly added missing types, methods, and fields:

- The *DataReader* operation **acknowledge_sample** with **ack_response_data** was missing and has been added.
- **dds.Presentation.drop_incomplete_coherent_set** was missing and has been added.
- **dds.DomainParticipant** - the following methods have been added: **discovered_participant_subject_name, discovered_participants_from_subject_name, banish_ignored_participants**.
- **dds.DomainParticipantQos** - the following QoS policies have been added: **partition, default_unicast**.
- **dds.BuiltinTopicReaderResourceLimits** was missing **max_fragmented_samples_per_remote_writer**, which has now been added.
- The constant dds.DataReaderResourceLimits.AUTO_MAX_TOTAL_INSTANCES was missing and has been added.
- **dds.DataWriterProtocol.initial_virtual_sequence_number** was missing and has been added.
- **dds.DiscoveryConfigBuiltinChannelKindMask** was missing and has been added.
- **dds.DomainParticipantResourceLimits.serialized_type_object_dynamic_allocation_threshold** was missing and has been added.
- The constant dds.PublishMode.PUBLICATION_PRIORITY_UNDEFINED was missing and has been added.
- **dds.SystemResourceLimits.initial_objects_per_thread** was missing and has been added.
- **dds.DataWriterCacheStatus** was missing the following read-only properties, which have been added: **alive_instance_count, alive_instance_count_peak, disposed_instance_count, disposed_instance_count_peak, unregistered_instance_count, unregistered_instance_count_peak**.

- **dds.CompressionSettings** was missing the following constants, which have been added: COMPRESSION_LEVEL_DEFAULT, COMPRESSION_LEVEL_BEST_SPEED, COMPRESSION_LEVEL_BEST_COMPRESSION.
- **dds.Cookie** was missing a no-argument constructor, which has been added.
- **dds.AcknowledgmentInfo.cookie** was missing and has been added.
- The constant dds.FlowControllerProperty.DEFAULT_FLOW_CONTROLLER_NAME was missing and has been added.
- **dds.Property** can now be created from a dictionary.

4.8.5.3.4 Other

- In Entity types, listener is now a read-only property; use **set_listener** to change it with a status mask.
- The *DataReader* read/take operations include several changes. See RTI Connext Core Libraries What's New in 7.1.0.
- **dds.GroupData's** constructor did not initialize the bytes correctly and has been fixed.
- Setting **dds.EntityName.name** and **role_name** to None explicitly was not supported and caused a crash. This has been fixed.

[RTI Issue ID PY-85]

4.8.5.4 Possible deadlock between creation of a dds.Topic and a listener callback

A possible deadlock could have occurred, leaving the Python interpreter hanging indefinitely when a **dds.Topic** was created at the same time as a listener callback was in process. This problem has been resolved.

[RTI Issue ID PY-88]

4.8.5.5 Listeners may not have been called in some situations

Entity listener callbacks may not have been called in some situations, causing the application to miss notifications about Entity status changes. This problem was due to a bug in pybind11 version 2.8.1. The build instructions have been updated to require pybind11 2.9.0, which solves this problem.

[RTI Issue ID PY-92]

4.9 Fixes Related to XML Configuration

4.9.1 Memory leak after an error parsing XML file with <include> tag

If the user's application failed to parse an XML file containing an <include> tag, this caused a memory leak. For example:

```
<types>
<include file=""myFile.xml"">

<struct name=""MyStruct"">
    <member name=""m1"" type=""unknownType"" />
</struct>

</types>
```

This file cannot be parsed because **m1** refers to an unknown type. When the application finished, running a memory profiling tool such as Valgrind^{TM} showed there was a memory leak. This problem has been resolved.

[RTI Issue ID CORE-12831]

4.9.2 Failed to parse XML configuration file containing type member with useVector attribute

Connext libraries failed to parse XML files containing a type member with the attribute useVector, although this is a legal attribute.

For example:

```
<types>
    <struct name= "MyType">
        <member name="m1" sequenceMaxLength="100" useVector="true" type="int32"/>
    </struct>
</types>
```

Parsing this file failed with the following error:

```
RTIXMLParser_validateOnStartTag:Parse error at line xxx: Unexpected attribute 'useVector'
```

This problem has been fixed.

[RTI Issue ID CORE-12949]

4.9.3 XML composition overwrote system information properties with defaults instead of correct values

The XML composition mechanism (described in "QoS Profile Inheritance and Composition" in the *QoS Profiles* section of the "Configuring QoS with XML" chapter of the RTI Connext Core Libraries User's Manual) had an issue with the way system properties (described in "System Properties" in the *Working* *with DDS Domains* chapter of the RTI Connext Core Libraries User's Manual) set in an XML Snippet were applied to a <domain_participant_qos> in an XML Profile referencing the Snippet. The properties set in the XML Snippet were not applied to the <domain_participant_qos>, which ended up using the automatic values generated by *Connext*.

Here is an example that illustrates the problem:

```
<qos_library name="SampleQoSLib">
    <qos_profile name="ParentProfile">
        <domain_participant_qos>
            <property>
                <value>
                    <element>
                        <name>dds.sys_info.hostname</name>
                        <value>CustomHostName</value>
                    </element>
                </value>
            </property>
        </domain_participant_qos>
    </qos_profile>

    <qos_profile name="ChildProfile" is_default_qos="true">
        <domain_participant_qos>
            <base_name>
                <element>SampleQosLib::ParentProfile</element>
            </base_name>
            <property>
                <value>
                    <element>
                        <name>dds.sys_info.username</name>
                        <value>CustomUserName</value>
                    </element>
                </value>
            </property>
        </domain_participant_qos>
    </qos_profile>
</qos_library>
```

The <domain_participant_qos> in the ChildProfile ended up with the following values for the system information properties:

- dds.sys_info.hostname - The default value rather than the CustomHostName value as set in the <domain_participant_qos> in ParentProfile, because of the overwriting problem described above.
- dds.sys_info.username - The set value of CustomUserName, which is the correct value.

This issue has been resolved.

[RTI Issue ID CORE-13090]

4.10 Fixes Related to Request-Reply and RPC

4.10.1 RPC interface evolution did not work

Remote Procedure Call (RPC) interfaces were designed to be extensible. A service and a client can communicate even when they have a different number of interfaces. For example:

A base service definition in IDL could be as follows:

```
@service
interface RobotControl {
    Coordinates walk_to(Coordinates destination, float speed);
    float get_speed();
};
```

If you add new operations to the service interface, such as the following:

```
@service
interface RobotControl {
    Coordinates walk_to(Coordinates destination, float speed);
    float get_speed();
    float get_position();
};
```

Or remove operations from the service interface, such as the following:

```
@service
interface RobotControl {
    Coordinates walk_to(Coordinates destination, float speed);
};
```

They should remain interoperable.

However, in the previous release, the service and the client wouldn't communicate in any case.

This problem has been resolved. A client can now invoke an operation in a service with more or fewer operations. If the operation exists in the service, it will receive a valid response. If the operation doesn't exist in the service, the service will respond accordingly and the client will throw the standard exception **dds::rpc::RemoteUnknownOperationError**.

[RTI Issue ID REQREPLY-105]

4.10.2 Exceptions sending result of remote operation may have crashed server application

In an RPC server-side application, the user implements the functional interface. The Server uses a thread pool to call those functions with the input sent from the client (Request) and obtain the result. The result is then sent to the client (Reply). The Reply is automatically written using a DDS *DataWriter*. If the **write()** operation failed, the resulting exception would crash the current thread in the thread pool and possibly crash the entire server-side application (a typical **write()** exception is a

Timeout). Since the Reply is sent by the server from a separate thread, the user application has no way of catching the exception or sending the Reply again.

This problem has been resolved. If an exception occurs, it is caught and logged. The Reply is never sent. User applications have two ways to react to this event:

- The server application can install a **rti::config::Logger::output_handler** to monitor errors.
- The client application will see a timeout in the function call. The application can then react accordingly (e.g., calling the function again later).

[RTI Issue ID REQREPLY-111]

4.10.3 RPC: deadlock when Server::close() was called before Server::run()

In the unlikely scenario that a Server was created and then closed before running (the method **Server-::close()** was called before **Sever::run()**), **run()** would never return unless a timeout was specified. This problem has been resolved.

[RTI Issue ID REQREPLY-113]

4.10.4 Possible unbounded memory growth when creating many Requesters

This issue was fixed in release 7.0.0, but not documented at that time.

When a Requester is created, a ContentFilteredTopic is internally created on the Requester's *DomainParticipant*. This ContentFilteredTopic is exclusively created for each Requester and was never deleted until the *DomainParticipant* was deleted. This may have caused applications that continuously create and delete Requesters on the same *DomainParticipant* to see unbounded memory growth.

This problem has been resolved in all language APIs. The Requester destructor or deletion function now deletes its ContentFilteredTopic.

[RTI Issue ID REQREPLY-35]

4.10.5 Memory leak in Java Request-Reply API

This issue was fixed in release 7.0.0, but not documented at that time.

The Java Request-Reply API leaked a small amount of native heap memory every time a Requester was created. The leak was caused by a few internal WaitSet objects, which did not have a finalizer and were not explicitly deleted either.

[RTI Issue ID REQREPLY-94]

4.10.6 Possible data race using Sample and WriteSample classes (Traditional C++ API only)

This issue was fixed in release 7.0.0, but not documented at that time.

The Sample and WriteSample classes are wrapper classes in the Traditional C++ Request-Reply API that used to initialize the underlying user data lazily: the data was initialized the first time it was accessed with the **data()** member function.

This approach made the access to the data unsafe. A data race could occur when two or more threads competed to access the same sample object for the first time. This problem has been resolved. The lazy approach has been reversed, and the data is now initialized in the constructor.

[RTI Issue ID REQREPLY-95]

4.11 Fixes Related to Shipped Examples

4.11.1 Hello World TCP example always linked TCP Transport library dynamically

The C **hello_world_tcp** example always linked the *RTI TCP Transport* library dynamically, even if you wanted to use static linking. This issue has been fixed. Now, the **nddstransporttcp** library is linked statically unless you choose Debug DLL or Release DLL from the configuration pull-down menu of the provided projects on Windows. Or, when using a makefile, the *TCP Transport* library is now linked statically, unless you pass the "SHAREDLIB=1" argument to the make command.

Furthermore, the README file for the example has been updated with further instructions on what additional libraries need to be added to the makefile or project file when TLS is enabled.

[RTI Issue ID COREPLG-577]

4.12 Fixes Related to Vulnerabilities

4.12.1 Arbitrary read access while parsing malicious RTPS message

Arbitrary read access could occur while parsing a malicious RTPS message. This issue has been fixed.

4.12.1.1 User Impact without Security

A vulnerability in the *Connext* application could have resulted in the following:

- Arbitrary read access while parsing a malicious RTPS message.
- Remotely exploitable.
- Potential impact on confidentiality of *Connext* application.

- CVSS Base Score: 8.2 HIGH
- CVSS v3.1 Vector: AV:N/AC:L/PR:N/UI:N/S:U/C:L/I:N/A:H

4.12.1.2 User Impact with Security

Same impact as described in "User Impact without Security," above.

[RTI Issue ID CORE-13160]

4.12.2 Out-of-bounds read while parsing malicious RTPS message

An out-of-bounds read could occur while parsing a malicious RTPS message. This issue has been fixed.

4.12.2.1 User Impact without Security

A vulnerability in the *Connext* application could have resulted in the following:

- Out-of-bounds read while parsing a malicious RTPS message.
- Remotely exploitable.
- Potential impact on confidentiality of *Connext* application.
- CVSS Base Score: 6.5 MEDIUM
- CVSS v3.1 Vector: AV:N/AC:L/PR:N/UI:N/S:U/C:L/I:N/A:L

4.12.2.2 User Impact with Security

Same impact as described in "User Impact without Security," above.

[RTI Issue IDs CORE-13240 and CORE-13264]

4.12.3 Out-of-bounds write while parsing malicious RTPS message

An out-of-bounds write could occur while parsing a malicious RTPS message. This issue has been fixed.

4.12.3.1 User Impact without Security

A vulnerability in the *Connext* application could have resulted in the following:

- Out-of-bounds write while parsing a malicious RTPS message.
- Remotely exploitable.
- Potential impact on integrity of *Connext* application.

- CVSS Base Score: 8.2 HIGH
- CVSS v3.1 Vector: AV:N/AC:L/PR:N/UI:N/S:U/C:N/I:L/A:H

4.12.3.2 User Impact with Security

Same impact as described in "User Impact without Security," above.

[RTI Issue ID CORE-13279 and CORE-13150]

4.12.4 Buffer overflow in shared memory if memory was tampered

A buffer overflow occurred when publishing or receiving metadata or data over a tampered shared memory segment. This issue has been fixed.

4.12.4.1 User Impact without Security

- Exploitable from the same node the *Connext* application is running (needs access to shared memory segment).
- Application crash. Potential impact to the integrity or confidentiality of the *Connext* application.
- CVSS Base Score: 7.8 HIGH
- CVSS v3.1 Vector: AV:L/AC:L/PR:L/UI:N/S:U/C:H/I:H/A:H

4.12.4.2 User Impact with Security

Same impact as described in "User Impact without Security," above.

[RTI Issue ID CORE-13300]

4.12.5 Out-of-bounds read while uncompressing malformed data from malicious RTPS message

An out-of-bounds read occurred while uncompressing malformed data from a malicious RTPS message. This issue has been fixed.

4.12.5.1 User Impact without Security

A vulnerability in the *Connext* application could have resulted in the following:

- Out-of-bounds read while uncompressing malformed data from a malicious RTPS message.
- Remotely exploitable.
- Potential impact on confidentiality of *Connext* application.

- CVSS Base Score: 4.8 MEDIUM
- CVSS v3.1 Vector: AV:N/AC:H/PR:N/UI:N/S:U/C:L/I:N/A:L

4.12.5.2 User Impact with Security

Same impact as described in "User Impact without Security," above.

[RTI Issue ID CORE-13548]

4.13 Fixes Related to Crashes

4.13.1 Rare segmentation fault when deleting DomainParticipant or Publisher containing DataWriters using durable writer history

A *Connext* application may have crashed after deleting a *DomainParticipant* or *Publisher* containing *DataWriters* using durable writer history. This issue has been fixed.

[RTI Issue ID CORE-12297]

4.13.2 Segmentation fault when creation of DomainParticipant failed due to lack of resources

An application may have produced a segmentation fault using the release libraries if the creation of a *DomainParticipant* failed because the following resource limit was exceeded: **participant_factory_ qos.resource_limits.max_objects_per_thread**.

With debug libraries, you may have seen a precondition error such as this:

```
FATAL U000000011d1a15c0_ [CREATE
DP|LC:DISC]Mx06:/connextdds/event.1.0/srcC/activeDatabase/ActiveDatabase.c:275:RTI0x2000027:
!precondition
```

This problem has been fixed.

[RTI Issue ID CORE-12654]

4.13.3 Potential hang upon SIGSEGV signal from a Connext application

For debuggability purposes, *Connext* applications log a backtrace when a SIGSEGV signal is triggered.

In previous releases, this feature may have triggered a hang during the logging of the backtrace. In this release, we address this issue by disabling the logging of the backtrace by default in release libraries (but still keeping it enabled for debug libraries).

This default behavior can be modified by setting the new *DomainParticipant*-level property **dds.-participant.enable_backtrace_upon_sigsegv**. See “New property to manually enable or disable logging backtrace upon SIGSEGV signal from a Connext application” in RTI Connext Core Libraries What's New in 7.1.0.

[RTI Issue ID CORE-12794]

4.13.4 Creating DynamicDataTypePlugin with TypeCode from discovery and using content filtering caused segmentation fault

If the TypeCode that was received from endpoint discovery data (**PublicationBuiltinTopicData.type_code or SubscriptionBuiltinTopicData.type_code**) was used to create a DynamicDataTypeSupport in an application that was also using ContentFilteredTopics and setting **ResourceLimitsQosPolicy.type_code_max_serialized_length** to a non-zero value, the application issued a segmentation fault.

ResourceLimitsQosPolicy.type_code_max_serialized_length is 0 by default, which avoids the segmentation fault.

This issue has been fixed.

[RTI Issue ID CORE-12992]

4.13.5 Application crash when calling DDS_DataReader_take_discovery_snapshot on a DataReader with a ContentFilteredTopic

When taking a discovery snapshot by calling the **DDS_DataReader_take_discovery_snapshot** function on a *DataReader* with a ContentFilteredTopic, the application crashed when trying to obtain non-valid *DomainParticipant* information. This issue has been fixed. Now, *DomainParticipant* information is obtained correctly for *DataReaders* with ContentFilteredTopics.

[RTI Issue ID CORE-13011]

4.13.6 Crash with NULL listeners and non-none status masks in C applications that mixed types with and without Zero Copy

In a C application, a crash occurred when both of these were true:

- Types with and without Zero Copy transfer over shared memory were mixed inside the same DomainParticipantFactory instance.
- A *DataReader* or *DataWriter* of the non-Zero Copy types had a NULL listener and a **DDS_StatusMask** different than DDS_STATUS_MASK_NONE.

The crash occurred because *Connext* invoked a NULL listener callback for the statuses enabled in the endpoints' **DDS_StatusMask**.

When there is a Zero Copy type inside an application, some extra pre-processing related to Zero Copy is done before creating the endpoints and setting the listeners. In that extra pre-processing, for non-Zero Copy types, the NULL listener was incorrectly replaced with a non-null listener object with all its callbacks set to NULL. Then, *Connext* was not checking if the callbacks were NULL before calling them (the listener consistency is checked before the incorrect replacement; therefore, at that point, it was assumed the listener object was consistent).

This issue is fixed. The listener is no longer replaced with an invalid listener object, and *Connext* will always check if the callbacks are NULL before calling them.

[RTI Issue ID CORE-13151]

4.13.7 Memory was read after it was freed by deleting a Topic with local logging level enabled

If the local logging level was enabled while deleting a topic, *Connext* would use recently freed memory from the deleted *Topic* to print a log message. Using the recently freed memory could cause a crash if local logging was enabled. A log message is now printed immediately before the *Topic* is deleted, so the possibility of using freed memory is eliminated.

[RTI Issue ID CORE-13226]

4.13.8 Possible segmentation fault when disabling loopback interface

When a previously enabled loopback interface on a host computer was disabled, a segmentation fault could occur. The handling of loopback interfaces has been redesigned to remove this possibility.

[RTI Issue ID CORE-13228]

4.13.9 Segmentation fault could occur if creation of DataReader failed

In some cases, a segmentation fault would occur if the creation of a *DataReader* failed. This problem has been fixed.

[RTI Issue ID CORE-13387]

4.13.10 Potential crash when DomainParticipant deleted after creating DataWriter with automatic liveliness kind

There was a small possibility of a crash occurring when the *DomainParticipant* was deleted immediately after creating a *DataWriter* with an AUTOMATIC_LIVELINESS_QOS **kind** in the LIVELINESS QoS policy. This problem has been resolved.

[RTI Issue ID CORE-13524]

4.13.11 Possible crash on TCP transport when large number of file descriptors were open

A *Connext* application that used the TCP transport and was built using **_FORTIFY_SOURCE**, which is set by default by some operating systems, could crash if one of the sockets for TCP had a file descriptor higher than FD_SETSIZE (1024). This issue has been fixed. Now, *Connext* overwrites the value of FD_SETSIZE, allowing an application using the TCP transport to open up to 32768 file descriptors, except on Android, where it is not possible to overwrite this value.

[RTI Issue ID COREPLG-644]

4.13.12 Application using Monitoring Libraries may have produced segmentation fault during DataReader creation

In 6.0.x releases and above, an application using the Monitoring Library may have produced a segmentation fault during *DataReader* creation. The issue was very rare and only occurred if a *DataReader* received a sample immediately after being enabled. This issue has been fixed.

[RTI Issue ID MONITOR-429]

4.13.13 Possible segmentation fault when using Monitoring Library

When using monitoring libraries, a rare race condition may have led to a segmentation fault. This issue was more likely to occur if the *Connext* application using the monitoring libraries created and deleted entities often. This problem has been resolved.

> **Note:** This problem was reported as fixed in MONITOR-252, in release 6.0.1; however, that fix did not apply to *Publishers* and *Subscribers*. This fix protects applications when frequently creating and deleting *Publisher* or *Subscriber* entities as well.

[RTI Issue ID MONITOR-516]

4.14 Other Fixes

4.14.1 Error sending batch when batch size exceeded transport MTU

A *DataWriter* configured to use batching may have failed to send a batch to the destination addresses associated with a transport (e.g, UDPv4) if the batch size exceeded the **message_size_max** (MTU) of the transport.

This problem has been resolved. Now, the batch is automatically flushed when exceeding the minimum **message_size_max** across all installed transports.

[RTI Issue ID CORE-2639]

4.14.2 Broken communication when DataWriter with transport priority discovered DataReader with multicast receive address

If a *DataWriter* that had a non-default **DataWriterQos.transport_priority** value set discovered a *DataReader* with a multicast receive address, the *DataWriter* and any other *DataWriters* within the same participant were not able to send any traffic over unicast. This could cause communication failures in a number of different scenarios, including a broken reliability protocol due to the inability to send heartbeats over unicast or the inability to communicate with other *DataReaders* that have not been configured to use a multicast receive address.

This issue was introduced in 6.1.0. This issue has been resolved.

[RTI Issue ID CORE-12772]

4.14.3 Potential hang upon SIGSEGV signal from a Connext application

For debuggability purposes, *Connext* applications have the ability to log a backtrace when a SIGSEGV signal is triggered.

In previous releases, this feature may have triggered a hang during the logging of the backtrace. In this release, we address this issue by disabling the logging of the backtrace in release libraries (but still keeping it enabled for debug libraries).

This default behavior can be modified by setting a new participant-level property, **dds.-participant.enable_backtrace_upon_sigsegv**. The accepted values for this new property are: "auto" for the default behavior (backtrace only enabled in debug libraries), "true" for enabling the logging of the backtrace, and "false" for disabling it.

Note: This property takes effect upon the creation of the first *DomainParticipant* within a process. Consequently, if a SIGSEGV signal is received before the creation of the first *DomainParticipant*, the default behavior will be applied (backtrace enabled in debug libraries and disabled in release libraries).

[RTI Issue ID CORE-12794]

4.14.4 No more than 100 asynchronous publisher threads could be created

A change to the thread naming convention inadvertently limited the number of asynchronous publisher threads to 100. The limit is now 65,536. These limits also apply to receive threads, asynchronous waitset threads, and persistence service threads.

[RTI Issue ID CORE-12874]

4.14.5 Potential memory leak when creation of any of the built-in discovery plugins failed

The first time a *DomainParticipant* is created in an application, some memory is allocated globally for each of the builtin discovery plugins (SPDP and SEDP) enabled for that *DomainParticipant*. This global memory is released when finalizing the DomainParticipantFactory instance.

However, if there was a failure in the creation of any of the builtin discovery plugins during the *DomainParticipant* creation, the *DomainParticipantFactory* was not notified properly that this global memory was allocated. Therefore, finalizing the *DomainParticipantFactory* instance did not release the memory, causing a leak.

This problem is fixed. Finalizing the *DomainParticipantFactory* instance always releases the memory if it was previously allocated, regardless of whether or not a failure occurred.

[RTI Issue ID CORE-12882]

4.14.6 Samples could be lost using group order access or collaborative DataWriters

There was a possibility of *DataReader* queue corruption, when using group order access or collaborative *DataWriters*, that may have provoked the *DataReader* to stop receiving samples. The possibility was very small and may have occurred randomly since it was caused by an uninitialized flag.

[RTI Issue ID CORE-13153]

4.14.7 Unexpected precondition error while creating a DomainParticipant with debugging libraries using fast database cleanup period

You may have seen the following precondition error while creating a *DomainParticipant* with debugging libraries if **participant_qos.database.cleanup_period** was updated to a small value.

```
FATAL rCo96144####Dtb Mx0D:/rti/jenkins/workspace/connextdds_ci_fastbuild-debug_
develop/pres.1.0/srcC/participant/Participant.c:3102:RTI0x200003b:!precondition: "me->_
service == ((void *)0)"
```

Release libraries did not have this issue.

This problem has been fixed.

[RTI Issue ID CORE-13204]

4.14.8 Release 6.1.2 was not FACE compliant

The *Connext* 6.1.2 release was not FACE compliant due to usage of the realpath system call. This problem has been resolved.

[RTI Issue ID CORE-13340]

4.14.9 Problems visualizing participants using Generic.MinimalMemoryFootprint profile with Admin Console

RTI Admin Console could not correctly visualize *DomainParticipants* using the **Generic.MinimalMemoryFootprint** profile. Some of the information, such as process ID and host name, was invalid. This problem has been fixed.

[RTI Issue ID CORE-13509]

4.14.10 Using dh_param_files leaked memory

Using the property **tls.cipher.dh_param_files** leaked memory when deleting the *DomainParticipant*. A memory checking tool, such as valgrind, would have reported the leak in the OpenSSL function **PEM_read_bio_DHparams**, which is called by the RTI function **RTITLS_tmp_dhparam_callback**. This problem only affected applications using OpenSSL 1.0.2 or applications communicating with applications using OpenSSL 1.0.2. For example, *TLS Support* 5.3 uses OpenSSL 1.0.2, but version 7.0.0 of *TLS Support* could still communicate with version 5.3, so the leak could also happen in version 7.0.0.

This problem has been fixed; memory will no longer be leaked in this scenario. For example, if *TLS Support* 7.1.0 communicates with an application using OpenSSL 1.0.2, the leak will not occur.

[RTI Issue ID COREPLG-641]

4.14.11 Failure to load a string-based private key leaked memory

If you set the property **tls.identity.private_key** or **tls.identity.rsa_private_key**, and you either specified a wrong or missing value for the property **tls.identity.private_key_password** or specified a malformed private key, then memory would be leaked upon *DomainParticipant* creation failure. A memory checking tool, such as valgrind, would report the leak in the OpenSSL function **BIO_new_mem_buf**, which is called by the RTI function **RTITLS_context_init**.

This problem has been fixed. Memory will no longer be leaked in this scenario.

[RTI Issue ID COREPLG-643]

4.14.12 Incorrect "Supported platforms" documentation section for FindRTIConnextDDS.cmake

Now the documentation section in the "FindPackage" script (**FindRTIConnextDDS.cmake**) file listing the "Supported platforms" matches the *RTI Connext Core Libraries Platform Notes*.

[RTI Issue ID INSTALL-548]

4.14.13 CONNEXTDDS_ARCH environment variable in FindPackage script was not picked up correctly

Previously, only the CONNEXTDDS_ARCH CMake variable in the "FindPackage" script (**FindRTIConnextDDS.cmake**) could be used to define the *Connext* official architecture to use. Now, the environment variable with the same name can be used, too.

[RTI Issue ID INSTALL-691]

4.14.14 In FindPackage script, low_bandwidth_edisc imported target library was missing

In the "FindPackage" script (**FindRTIConnextDDS.cmake**), the **low_bandwidth_edisc** imported target library was missing, incorrectly named **low_bandwidth_discovery_static**. When you tried to link against **low_bandwidth_discovery_static**, the script actually linked against the LOW_BANDWIDTH_EDISC libraries. And you couldn't link against **low_bandwidth_edisc** because there was no imported target with that name.

In the following example, the second TARGET should have been called **low_bandwidth_edisc**:

```
######################## Low bandwidth plugins #########################
    # Discovery Static
    create_connext_imported_target(
        TARGET "low_bandwidth_discovery_static"
        VAR "LOW_BANDWIDTH_DISCOVERY_STATIC"
        DEPENDENCIES
            RTIConnextDDS::c_api
    )

    # EDISC
    create_connext_imported_target(
        TARGET "low_bandwidth_discovery_static"
        VAR "LOW_BANDWIDTH_EDISC"
        DEPENDENCIES
            RTIConnextDDS::c_api
    )
```

This problem has been fixed.

[RTI Issue ID INSTALL-719]

4.14.15 Segmentation fault when mixing build types in applications linked against libraries in "Find Package" Cmake script

Mixing Release and Debug build types in applications linked against *Connext* libraries in the "Find Package" script (**FindRTIConnextDDS.cmake**) could lead to undesired behaviors like double-freeing pointers, once for the Debug symbol and once for the Release symbol, and in the end causing the application to abort.

The new CONNEXT_LIBS_BUILD_TYPE CMake variable has been added to control the *Connext* libraries build type (Release/Debug). This variable will allow three values: Auto, Release, and Debug.

By default (the Auto value), **FindRTIConnextDDS.cmake** will populate the IMPORTED_LOCATION_DEBUG and IMPORTED_LOCATION_RELEASE properties of all the *Connext* imported target libraries. This means that the *Connext* libraries will be provided in the same build type as the global build (specified by the CMAKE_BUILD_TYPE value).

If you provide Release or Debug values to the CONNEXT_LIBS_BUILD_TYPE variable, the script will force populating only the IMPORTED_LOCATION property of the *Connext* imported target libraries. So, regardless of the CMAKE_BUILD_TYPE value, the *Connext* libraries will have the build type given in the CONNEXT_LIBS_BUILD_TYPE variable.

[RTI Issue ID INSTALL-793]

Chapter 5 What's Fixed in 7.0.0

This section describes bugs fixed in *Connext* 7.0.0. These fixes have been made since 6.1.1 was released.

5.1 Fixes Related to Callbacks and Waitsets

5.1.1 Unsafe combinations of masks and Listeners may have led to segmentation fault

When entities are created, a *Listener* may be provided by the user to receive calls when specified events occur. The events of interest are set using a **StatusKind** mask. If an event set in the **StatusKind** mask occurs, but no callback function has been assigned by the user, a null pointer dereference will occur. *Connext* checks for many of these errors and prevents the creation of entities when this error is present. However, some of these cases were not checked, allowing unsafe combinations of masks and *Listeners* to be used. This problem has been resolved. The new, stricter checking may cause entity creation errors when no errors were detected before.

[RTI Issue ID CORE-12610]

5.1.2 Failure calling DDS_Subscriber::get_datareaders in DDS_SubscriberListener::on_data_on_readers callback implementation

You may have seen the following errors when invoking **DDS_Subscriber::get_datareaders()** within the implementation of the **DDS_SubscriberListener::on_data_on_readers()** callback:

```
ERROR [0x01011B2D,0x8A450DE1,0xBAE5A2A0:0x80000009|SET GROUP LISTENER|GET READERS]
REDACursor_modifyReadWriteArea:!freeze read write area
ERROR [0x01011B2D,0x8A450DE1,0xBAE5A2A0:0x80000009|SET GROUP LISTENER|GET READERS]
PRESPsReaderGroup_getEA:!modify pres psReaderGroup
ERROR [0x01011B2D,0x8A450DE1,0xBAE5A2A0:0x80000009|SET GROUP LISTENER|GET READERS]
PRESPsReaderGroup_lock:!take semaphore
ERROR [0x01011B2D,0x8A450DE1,0xBAE5A2A0:0x80000009|SET GROUP LISTENER|GET READERS]
PRESPsReaderGroup_beginGetPsReaders:!get PRESPsReaderGroup_lock
```

```
ERROR [0x01011B2D,0x8A450DE1,0xBAE5A2A0:0x80000009|SET GROUP LISTENER|GET READERS}
DDS_Subscriber_begin_get_datareadersI:ERROR: Failed to get PRESPsReaderGroup_
beginGetPsReaders
ERROR [0x01011B2D,0x8A450DE1,0xBAE5A2A0:0x80000009|SET GROUP LISTENER|GET READERS]
DDS_Subscriber_get_datareaders:ERROR: Failed to get DDS_Subscriber_begin_get_datareaders
```

In addition, when using the Traditional C++ API and the legacy .NET API, the application generated a segmentation fault after printing the error. The problem occurred only when:

- You installed a *Listener* on the *Subscriber* using the API **DDS_Subscriber::set_listener()** after the *Subscriber* was enabled.
- Or, you installed a *Listener* on the *DomainParticipant* using the API **DDS_Participant::set_listener()** after the *DomainParticipant* was enabled. This problem has been resolved.

[RTI Issue ID CORE-12316]

5.1.3 DDS_SubscriberListener::on_data_on_readers on a participant or subscriber not called when Listener installed after the entity is enabled

The callback **DDS_SubscriberListener::on_data_on_readers()** was not invoked when there was data available, if these two conditions were met:

- The *Listener* callback **on_data_on_readers()** was installed after the *Subscriber* or *DomainParticipant* implementing it was enabled.
- The *Listener* callback **on_data_available()** was not installed at any level (*DomainParticipant, Publisher*, or *DataReader*).

This problem has been resolved.

[RTI Issue ID CORE-12338]

5.1.4 Unable to assign callback function for on_sample_removed event using Modern C++ API

You may have been unable to assign a callback function for the on_sample_removed event using the Modern C++ API. Support for this callback has been added to the Modern C++ API in this release.

[RTI Issue ID CORE-12646]

5.1.5 Using certain callbacks at DomainParticipant or Publisher level may have led to segmentation fault

Handlers were not correctly implemented for the **on_instance_replaced()**, **on_sample_removed()**, **on_application_acknowledgment()**, and **on_service_request_accepted()** callbacks at the

DomainParticipant and *Publisher* levels. This could have led to segmentation faults when the corresponding events were enabled. This problem has been resolved.

[RTI Issue ID CORE-12647]

5.2 Fixes Related to Discovery

5.2.1 Unexpected memory growth when DataReader could not be matched with DataWriter due to unexpected error condition

Failing to match a *DataReader* with a *DataWriter* because of unexpected error conditions may have led to unexpected memory growth, because *Connext* may not have cleaned up the resources associated with the remote match completely. This problem has been resolved.

[RTI Issue ID CORE-8257]

5.2.2 Possible crash upon discovery of applications with unreachable locators

If an application used **DDS_STATUS_MASK_ALL** for a *DomainParticipant* or *Publisher Listener* and an unreachable locator was discovered, the application enabling the *Listener* may have crashed. An unreachable locator occurs most commonly when a Subscribing application uses a transport that the Publishing application does not use. For example, the Publishing application could use UDPv4 and the Subscribing application could use both UDPv4 and UDPv6.

More rarely, a crash may have occurred when a pre-5.2.0 Subscribing application used the shared memory transport and a 5.2.0+ Publishing application was not using the UDPv6 transport. A log message was generated if both participants were running on the same machine and this condition occurred. This condition was caused by a change to the way that transports are identified starting in version 5.2.0.

[RTI Issue ID CORE-11818]

5.2.3 Communication problems with applications using shared memory on INTEGRITY systems

If an application on an INTEGRITY platform used the shared-memory transport, the *Connext* libraries sometimes incorrectly assessed that a shared-memory segment was stale and could be reclaimed, when in fact it was not stale. This situation caused problems with communication between *DomainParticipants*, since information could be sent to a shared-memory segment that did not get dequeued by the intended recipient.

You may have seen error messages like these and the application may have hung while deleting the *DomainParticipant*:

```
<Target Output> ERROR RTIOsapiSharedMemoryBinarySemaphore_take:OS WaitForSemaphore()
failure, error 0XD: ObjectClosed
<Target Output> ERROR NDDS_Transport_Shmem_receive_rEA:!take semaphore
```

```
<Target Output> ERROR RTIOsapiSharedMemoryBinarySemaphore_take:OS WaitForSemaphore()
failure, error 0X9: ObjectIsUseless
```

This problem has been resolved.

Incompatibility with 6.1.1 and prior releases:
The fix for this issue involved some changes that make shared-memory segments in applications incompatible with those in 6.1.1 (and earlier) versions.

[RTI Issue ID CORE-12097]

5.2.4 Types containing Typedefs were sent without the typedefs in discovery when using DynamicData

When an application was using a DynamicDataReader or DynamicDataWriter and using a type that contained a typedef, the type that was sent during endpoint discovery for that endpoint did not contain the typedef. While this did not cause any mismatches or communication failure, it did cause a number of issues that may have been noticeable depending on what other products you may have also been using.

See 5.2.5 Unbounded memory growth in Spy when discovering multiple endpoints with the same Topics and types below for details about the specific issues that you may have encountered. The *RTI Admin Console Release Notes* and *RTI Routing Service Release Notes* also have related information. (See ADMINCONSOLE-997 and ROUTING-971, respectively.)

This issue has been resolved, meaning that the exact type definition that is registered with the participant, containing typedefs, is sent during discovery. This is a change in behavior from 6.0.0-based applications, which sent the type definitions without the typedef information.

[RTI Issue ID CORE-12107]

5.2.5 Unbounded memory growth in Spy when discovering multiple endpoints with the same Topics and types

Each time *DDS Spy* discovered an endpoint, it unnecessarily made a copy of the TypeCode that was associated with the endpoint's *Topic*, leading to unbounded memory growth. This issue has been fixed.

[RTI Issue ID CORE-12136]

5.2.6 Unnecessary discovery traffic related to IP mobility events on interfaces irrelevant to the transport

When there is a change on a network interface (an IP mobility event), a *Connext* application will update and resend its discovery information to include these changes. The transport can consider a change on an interface irrelevant (for example, changes on interfaces in the **deny_interfaces_list** of the transport). In this case, the new discovery messages are exactly the same as announced before, generating unnecessary discovery traffic that could affect the performance of the application.

This problem has been fixed. Now *Connext* only updates and resends its discovery information if there was a change on an interface relevant to the transport.

[RTI Issue ID CORE-12664]

5.3 Fixes Related to Transports

5.3.1 Communication problems with applications using shared memory on INTEGRITY systems

If an application on an INTEGRITY platform used the shared-memory transport, the *Connext* libraries sometimes incorrectly assessed that a shared-memory segment was stale and could be reclaimed, when in fact it was not stale.

This situation caused problems with communication between *DomainParticipants*, since information could be sent to a shared-memory segment that did not get dequeued by the intended recipient.

You may have seen error messages like these and the application may have hung while deleting the *DomainParticipant*:

```
<Target Output> ERROR RTIOsapiSharedMemoryBinarySemaphore_take:OS WaitForSemaphore()
failure, error 0XD: ObjectClosed
<Target Output> ERROR NDDS_Transport_Shmem_receive_rEA:!take semaphore
<Target Output> ERROR RTIOsapiSharedMemoryBinarySemaphore_take:OS WaitForSemaphore()
failure, error 0X9: ObjectIsUseless
```

This problem has been resolved.

Incompatibility with 6.1.1 and prior releases:
The fix for this issue involved some changes that make the shared memory segments incompatible with those in 6.1.1 (and earlier) versions.

[RTI Issue ID CORE-12097]

5.3.2 Race condition could cause unbounded memory growth in TCP Transport Plugin

Due to a race condition, the TCP Transport Plugin may have leaked memory when creating a new connection if the creation happened at the same time the *DomainParticipant* was being destroyed. The cause of the leak was the TCP Transport Plugin reallocating memory that was already released by the *DomainParticipant*. The race condition was unlikely to happen. However, in a system that frequently creates and destroys entities (and, therefore, TCP connections) and that runs for long enough, it may have lead to unbounded memory growth. The issue has been resolved.

[RTI Issue ID COREPLG-618]

5.4 Fixes Related to Filtering and TopicQuery

5.4.1 Unnecessary repair traffic for DataWriters using TopicQueries and asynchronous publishing

Samples that are sent in response to a TopicQuery are directed to the *DataReader* that created that TopicQuery. This means that those samples are only sent to the *DataReader* that made the request and have that *DataReader's* GUID attached to each sample in the sample's metadata. All other *DataReaders* receive GAP protocol messages, indicating to them that a given sequence number or set of sequence numbers is not meant for them.

Due to a defect, when a *DataReader* sent a NACK message requesting some TopicQuery samples to be repaired, if the requested sequence numbers included samples that were meant for a different *DataReader*, the *DataWriter* did not filter these samples and resend a GAP message. Instead, the *DataWriter* sent the *DataReader* samples that were not meant for it and the *DataReader* had to filter these samples out itself. As a result, the *DataReaders* may have received samples that should have been filtered out on the *DataWriter* side, leading to an increase in network traffic.

The problem only affected repair traffic. When a sample was filtered out by the *DataWriter* because it was directed to a different *DataReader*, the *DataWriter* sent a GAP protocol message to the *DataReader*. If the GAP message was lost, the *DataReader* NACKed for the sample; instead of sending a new GAP message, the *DataWriter* sent the sample. This problem has been resolved.

[RTI Issue ID CORE-12589]

5.4.2 Connext application using filtering feature may have crashed after running out of memory

In release 6.1.1.2, a *Connext* application using filtering features (that is, ContentFilteredTopic, QueryConditions, or TopicQuery) may have crashed after running out of memory. This problem has been resolved.

[RTI Issue ID CORE-12661]

5.4.3 Unnecessary sample filtering on a DataReader for samples already filtered by a DataWriter

When doing writer-side filtering, a late-joining *DataReader* using a ContentFilteredTopic may have spent unnecessary CPU cycles evaluating samples that pass the ContentFilteredTopic's expression. When using writer-side filtering, the filter evaluation is done by the *DataWriter* and it should not be necessary for the *DataReader* to do it again on samples that pass the filter expression. This problem, which only occurred for late-joining *DataReaders*, has been fixed.

[RTI Issue ID CORE-11084]

5.4.4 Creation of a ContentFilteredTopic or reception of TopicQuery samples may have taken long time for complex types

The creation of a ContentFilteredTopic or reception of TopicQuery samples, may have taken a long time for complex types. This issue has been resolved.

[RTI Issue ID CORE-12179]

5.4.5 Continuous creation of TopicQueries may have led to unnecessary memory fragmentation in OS memory allocator

In releases 6.0.x and 6.1.x, the continuous creation of TopicQueries may have led to unnecessary memory fragmentation in the OS memory allocator of the applications that receive the TopicQuery requests and dispatch responses. This issue may have resulted in an unexpected increase of the resident set size (RSS) memory of the application receiving and dispatching the TopicQueries compared to previous *Connext* releases. This problem has been fixed.

[RTI Issue ID CORE-12352]

5.4.6 rti::topic::find_registered_content_filters led to infinite recursion

The function **rti::topic::find_registered_content_filters()** was incorrectly implemented and would lead to infinite recursion and stack overflow in any application that called it. This problem has been resolved. This function returns the names of previously registered custom content filters. It is a little-used feature and does not affect the commonly used SQL content filter.

[RTI Issue ID CORE-12512]

5.4.7 Incorrect results for Unions when using DynamicData or Content Filters

When using a DynamicDataReader, samples containing a union may have had incorrect or invalid data after deserialization if the *DataReader's* type contained members that were not present in the *DataWriter's* type and those members had non-zero default values.

When using content filters, the filter results may have been incorrect if the type contained a union and the filter expression filtered on fields within the union that were present in the *DataReader's* type but were not present in the *DataWriter's* type and those members had non-zero default values.

For example, see this *DataWriterType*:

```
struct innerStructPub {
    short shortMember;
};
@mutable
union ComplexUnionTypePub switch(long) {
    case 0:
        long longMember;
    case 1:
        innerStructPub structMember;
};
```

and this *DataReaderType*:

```
struct innerStructSub {
    short shortMember;
    @default(5) long longMemberWithDefault;
};
@mutable
union ComplexUnionTypeSub switch(long) {
    case 0:
        long longMember;
    case 1:
        innerStructSub structMember;
};
```

In the above types, the member **longMemberWithDefault** is only present in the *DataReader's* type and has a default value of 5, so any sample that is received from the *DataWriter* should have this value set to 5 when read from the *DataReader's* queue. Instead, the value was incorrectly 0 when using DynamicData.

In addition, if this member was used as part of a content filter expression, a *DataReader* always used the value of 0 instead of 5 when evaluating a sample from a *DataWriter* using the DataWriterType which could lead to incorrect filter results. These issues have been fixed.

[RTI Issue ID CORE-12517]

5.4.8 Samples may have been unnecessarily filtered by Connext DataReader when DataWriter was from different DDS vendor

A Connext *DataReader* using a ContentFilteredTopic unnecessarily evaluated its filter on samples coming from a different vendor *DataWriter* that already marked the samples as passing the *DataReader* filter. This issue may have led to an increase in CPU utilization on the *DataReader* side, but it did not affect functional correctness or bandwidth utilization.

The problem occurred because *Connext* was not compliant with the way a filter signature is calculated according to the Section 9.6.4.1, *Content filter info (PID_CONTENT_FILTER_INFO)*, in the Real-time Publish-Subscribe Protocol DDS Interoperability Wire Protocol (DDSI-RTPSTM) Specification version 2.5).

This problem has been resolved.

[RTI Issue ID CORE-12531]

5.5 Fixes Related to Group Presentation

5.5.1 Application may not have received samples of coherent set when using GROUP access scope and TRANSIENT_LOCAL durability

An application using **GROUP** access scope and **TRANSIENT_LOCAL** (or higher) durability may not have received the samples for some coherent sets, or it may have received the samples with delay.

Assume a coherent set '**CS1**' published by a set of *DataWriters* that are part of the same group. This coherent set was not provided to the application if all the following conditions were true:

1. The *DataReaders* receiving '**CS1**' matched with the *DataWriters* publishing '**CS1**' after the coherent set was published.
2. '**CS1**' did not contain samples for some of the *DataWriters* in the group, or the samples were removed after applying the Lifespan QoS Policy. If '**CS1**' contained at least one sample per *DataWriter* in the group, this problem did not occur.
3. The application did not publish a new coherent set after '**CS1**'; or, if it did, the new coherent set did not contain samples from at least one of the *DataWriters* that were missing samples from '**CS1**'.

If the third condition was not met, then the delivery of the coherent set would be delayed instead of the coherent set not being provided.

[RTI Issue ID CORE-12350]

5.5.2 Application may stop receiving samples from DataReaders using GROUP_PRESENTATION_QOS

An application may have stopped receiving samples from *DataReaders* that were part of a *Subscriber* using **GROUP_PRESENTATION_QOS** under the following scenario:

- The *Publisher's* group contained at least one keyed *DataWriter* and one unkeyed *DataWriter*.
- The *Subscriber's* group contained only keyed *DataReaders* or unkeyed *DataReaders*, but not both.

This problem has been resolved.

[RTI Issue ID CORE-12161]

5.5.3 Segmentation fault when using GROUP_PRESENTATION_QOS or HIGHEST_OFFERED_PRESENTATION_QOS and setting filter_redundant_samples to FALSE on DataReader

An application generated a segmentation fault if it created a *DataReader* with the following valid configuration:

- **subscriber_qos.presentation.access_scope = DDS_GROUP_PRESENTATION_QOS or DDS_HIGHEST_OFFERED_PRESENTATION_QOS**
- **datareader_qos.availability.max_data_availability_waiting_time = DDS_DURATION_ZERO**
- **datareader_qos.availability.max_endpoint_availability_waiting_time = DDS_DURATION_ZERO**
- **datareader_qos.property** contained **dds.data_reader.state.filter_redundant_samples** with the value “**false**”

This problem has been resolved by allowing the *DataReader* to be created.

[RTI Issue ID CORE-12771]

5.6 Fixes Related to XML Configuration

5.6.1 Parsing error loading XML configuration file containing a const whose expression refers to an enumerator

Connext failed to load an XML configuration file containing a const whose expression referred to an enumerator. For example:

```
<enum name="Enum1">
   <enumerator name="Enumerator1” value=”1”/>
</enum>
<const name="Const1" type="int32" value="Enumerator1+1"/>
```

Loading this XML failed with an error similar to this:

```
DDS_XMLConst_initialize:Parse error at line 10: type 'Enum1' is not typedef
```

This problem has been fixed.

[RTI Issue ID CORE-5553]

5.6.2 Discrepancy between range defined by schema and that defined by API

There were discrepancies between the ranges defined by the schema files and those defined by the API for certain elements. This problem has been resolved. Now, validating an XML against the XSD should not fail when setting a value that is inside the range as defined by the API.

[RTI Issue ID CORE-7099]

5.6.3 Parsing error loading XML configuration file with enum type containing enumerator whose value was an expression referring to a const

Connext failed to load an XML configuration file with an enum type containing an enumerator whose value was an expression referring to a const. For example:

```
<const name="Const1" type="int32" value="10"/>
<enum name="Enum1">
    <enumerator name="Enumerator1” value=”Const1”/>
</enum>
```

Loading this XML failed with an error similar to this:

```
DDS_XMLEnum_on_start_tag:Parse error at line xy: integer expected
```

This problem has been fixed.

[RTI Issue ID CORE-10060]

5.6.4 Parsing error loading an XML configuration file with enum type containing enumerator whose value was an expression

Connext failed to load an XML configuration file with an enum type containing an enumerator whose value was an expression. For example:

```
<enum name="Enum1">
   <enumerator name="Enumerator1” value=”1 + 1”/>
</enum>
```

Loading this XML failed with an error similar to this:

```
DDS_XMLEnum_on_start_tag:Parse error at line xy: integer expected
```

This problem has been fixed.

[RTI Issue ID CORE-10269]

5.6.5 Type limits not checked for some attributes of XML types definition

When XML was used for defining types (for example, when using DynamicData), type limits were not checked for some attributes. If the specified value for any of the attributes was too large or too small, a variable overflow occurred, leading to undefined behavior.

This problem is fixed. Type limits are checked, throwing a meaningful error when they are not met.

The affected attributes were as follows:

- **value** in union's **caseDiscriminator**. Valid values should be between -2147483648 and 2147483647.
- **sequenceMaxLength**. Valid values should be between 0 and 2147483647. -1 (unbounded) is also allowed.
- **stringMaxLength**. Valid values should be between 0 and 2147483647. -1 (unbounded) is also allowed.
- **arrayDimensions**. Valid values should be between 1 and 4294967295.

[RTI Issue ID CORE-12181]

5.6.6 Removed some elements in the XSD that were not supported internally but could be defined in XML

The following elements were configurable in XML although internally they are not supported:

Publisher QoS:

- **presentation.drop_incomplete_coherent_set**
- **asynchronous_publisher.thread.cpu_list**
- **asynchronous_publisher.thread.cpu_rotation**
- **asynchronous_publisher.asynchronous_batch_thread.cpu_list**
- **asynchronous_publisher.asynchronous_batch_thread.cpu_rotation**
- **asynchronous_publisher.topic_query_publication_thread.cpu_list**
- **asynchronous_publisher.topic_query_publication_thread.cpu_rotation**

Participant QoS:

- **discovery_config.publication_reader.min_app_ack_response_keep_duration**
- **discovery_config.subscription_reader.min_app_ack_response_keep_duration**
- **discovery_config.asynchronous_publisher.thread.cpu_list**
- **discovery_config.asynchronous_publisher.thread.cpu_rotation**
- **discovery_config.asynchronous_publisher.disable_asynchronous_batch**
- **discovery_config.asynchronous_publisher.asynchronous_batch_thread**
- **discovery_config.asynchronous_publisher.disable_topic_query_publication**
- **discovery_config.asynchronous_publisher.topic_query_publication_thread**

EventQosPolicy:

- **thread.cpu_list**
- **thread.cpu_rotation**

DatabaseQosPolicy:

- **thread.cpu_list**
- **thread.cpu_rotation**

Those elements have been removed from the XSD and are no longer configurable in XML.

[RTI Issue ID CORE-12366]

5.6.7 Builtin Discovery Plugins was not treated as a mask by the XSD file

Because of a bug in the XML Schema Definition (XSD), if you specified more than one value for the **DiscoveryConfigQosPolicy::builtin_discovery_plugins** mask, your XML editor reported that the expression was not valid when it should have been.

For example, according to the XSD, this expression was not allowed:

```
<domain_participant_qos>
    <discovery_config>
        <builtin_discovery_plugins>SPDP|SEDP</builtin_discovery_plugins>
    </discovery_config>
</domain_participant_qos>
```

This issue has been fixed, and the XSD now accepts expressions containing more than one Builtin Discovery Plugin. This issue occurred only while editing XML files because of the schema. If you ran an application with the above configuration, it did not fail.

[RTI Issue ID CORE-12740]

5.6.8 Parsing error loading an XML configuration file with an enum type containing an enumerator whose value was an expression referring to another enumerator

Connext failed to load an XML configuration file with an enum type containing an enumerator whose value was an expression using another enumerator. For example:

```
<enum name="Enum1">
    <enumerator name="Enumerator1” value=”0”/>
</enum>

<enum name="Enum2">
```

```
    <enumerator name="Enumerator2” value=”Enumerator1”/>
</enum>
```

Loading this XML would have failed with an error similar to this:

```
DDS_XMLEnum_on_start_tag:Parse error at line xy: integer expected
```

This problem has been fixed.

[RTI Issue ID CORE-12781]

5.7 Fixes Related to Vulnerabilities

5.7.1 Fixes related to Connext

This release fixes some potential vulnerabilities, including RTI Issue IDs CORE-12510 and CORE-12752.

5.7.2 Fixes related to third-party dependencies

This release fixes some potential vulnerabilities related to third-party dependencies, described below.

5.7.2.1 Potential crash or leak of sensitive information in Core Libraries XML parser due to vulnerabilities in Expat

The Core Libraries XML parser had a third-party dependency on Expat version 2.4.4, which is known to be affected by a number of publicly disclosed vulnerabilities.

These vulnerabilities have been fixed by upgrading Expat to the latest stable version, 2.4.8. See "Third-Party Software Upgrades" in RTI Connext Core Libraries What's New in 7.1.0.

The impact on *Connext* applications of using the previous version varied depending on your *Connext* application configuration:

- With *Connext Secure* (enabling RTPS protection):
  - Exploitable through a compromised local file system containing malicious XML/DTD files.
  - Could lead to arbitrary code execution.
  - CVSS v3.1 Score: 8.4 HIGH
  - CVSS v3.1 Vector: AV:L/AC:L/PR:N/UI:N/S:U/C:H/I:H/A:H
- Without *Connext Secure*:
  - Exploitable through a compromised local file system containing malicious XML/DTD files.

- Remotely exploitable through malicious RTPS messages.
- Could lead to arbitrary code execution.
- CVSS v3.1 Score: 9.8 CRITICAL
- CVSS v3.1 Vector: AV:N/AC:L/PR:N/UI:N/S:U/C:H/I:H/A:H

[RTI Issue ID CORE-12872]

5.7.2.2 Potential memory corruption when using Zlib compression due to vulnerability in Zlib

The user-data compression feature in the Core Libraries had a third-party dependency on Zlib version 1.2.11, which is known to be affected by a publicly disclosed vulnerability.

This vulnerability has been fixed by upgrading Zlib to the latest stable version, 1.2.12. See "Third-Party Software Upgrades" in RTI Connext Core Libraries What's New in 7.1.0.

The impacts on *Connext* applications of using the previous version were as follows:

- Exploitable by triggering the compression of a sample containing a malicious payload.
- The application could crash.
- CVSS v3.1 Score: 7.5 HIGH
- CVSS v3.1 Vector: AV:N/AC:L/PR:N/UI:N/S:U/C:N/I:N/A:H

[RTI Issue ID CORE-12877]

5.8 Fixes Related to APIs

5.8.1 Input parameters to Property and DataTag helper functions do not have "const"

In the C API, the following functions were incorrectly missing a **const** before the **policy** parameter:

- **DDS_PropertyQosPolicyHelper_lookup_property()**
- **DDS_PropertyQosPolicyHelper_lookup_property_with_prefix()**
- **DDS_PropertyQosPolicyHelper_get_properties()**
- **DDS_DataTagQosPolicyHelper_lookup_tag()**

This problem has been fixed. The policies are now "const" because these functions do not change the policy.

[RTI Issue ID CORE-3166]

5.8.2 Standard 64-bit integer types are now supported (Modern C++ API)

Previous releases of the Modern C++ API had platform-specific definitions for 64-bit integers, defined in **rti::core::int64** and **rti::core::uint64**. This was required to support certain pre-C++11 platforms.

This release redefines those two types as **std::int64_t** and **std::uint64_t**.

[RTI Issue ID CORE-10913]

5.8.3 Assigning DataWriter and DataReaderQos from a TopicQos caused a build error

DataWriterQos and DataReaderQos could not be constructed from a TopicQos assignment. You may have seen a compiler error such as:

```
error: conversion from 'TEntityQos<rti::topic::qos::TopicQosImpl>'  to
non-scalar type 'TEntityQos<rti::pub::qos::DataWriterQosImpl>' requested.
```

This problem has been resolved. Now this type of assignment works correctly. Any fields that are not in the TopicQos will use the default for the DataWriterQos or DataReaderQos.

[RTI Issue ID CORE-11185]

5.8.4 Copy of SampleInfo::coherent_set_info field was not supported

SampleInfo::coherent_set_info was not available when using take/read operations that did not loan the samples. The **SampleInfo::coherent_set_info** field was always set to NULL when you called the take/read operations that did not loan the samples. To get the **coherent_set_info** value, you had to use the read/take operations that loan the data.

In addition, the copy constructor and assignment operator in the Traditional C++ and Modern C++ APIs did not copy the **SampleInfo::coherent_set_info** field. This field was always set to NULL; it was your responsibility to make the copy and handle memory allocation and deletion for this field.

This problem has been fixed. If you work with the C API, starting with this release you will have to use the following functions to manipulate SampleInfo structures:

- **DDS_SampleInfo_initialize()**
- **DDS_SampleInfo_copy()**
- **DDS_SampleInfo_finalize()**

[RTI Issue ID CORE-11213]

5.8.5 In XML-based applications, generated IDL types did not take precedence over XML DynamicTypes (C# API)

In the C# API in previous releases, if a type was declared in XML as a dynamic type and also generated and registered by the application, the XML dynamic type took precedence. This led to the DataReaders or DataWriters using DynamicData instead of the generated C# user class. This behavior was unintuitive and inconsistent with the other language APIs. It has been resolved.

[RTI Issue ID CORE-11389]

5.8.6 Namespaces ignored when a type was explicitly registered in C# for XML-based applications

When a type was explicitly registered (this is only necessary to support generated IDL types with XML-Based Application Creation) as follows:

```
DomainParticipantFactory.RegisterType<A.B.Foo>()
```

The registered type name was to set to “Foo” instead of the expected “A::B::Foo”. In some situations, this may have stopped applications written in other languages to communicate with a C# application, if the regular algorithm of type matching was disabled.

[RTI Issue ID CORE-12074]

5.8.7 Corruption of LoanedDynamicData object when moved in some situations (Modern C++ API only)

Given a DynamicData sample, accessing a nested member within another nested member via **loan_ value()** and then moving the latter may have corrupted the former. For example, given a sample such that "my_sample.a.b" is a member of a constructed type (struct or union):

```
DynamicData my_sample(my_dynamic_type);
LoanedDynamicData loan1 = my_sample.loan_value(""a"");
LoanedDynamicData loan2 = loan1.get().loan_value(""b"");
// The following corrupts loan2
LoanedDynamicData loan1_moved = std::move(loan1);
```

This may have affected applications that explicitly move-constructed a double-nested LoanedDynamicData or that otherwise indirectly called the move constructor in this situation (for example, by resizing a std::vector of LoanedDynamicData elements).

The LoanedDynamicData's move constructor and move-assignment operators have been fixed.

[RTI Issue ID CORE-12272]

5.8.8 Calling DynamicData::set_complex_member with an aliased type failed

Calling **DynamicData::set_complex_member()** with an aliased type failed. For example, given the following types:

```
struct Foo {
long x;
long y;
};
typedef Foo TypedefFoo;
struct MyType {
Foo my_inner_struct;
TypedefFoo my_typedef_struct;
};
```

The following code should have worked to set the my_typedef_struct member:

```
DDS_DynamicData *data = DDS_DynamicData_new(
    MyType_get_typecode(),
    &DDS_DYNAMIC_DATA_PROPERTY_DEFAULT);
    DDS_DynamicData *inner_data = DDS_DynamicData_new(
    TypedefFoo_get_typecode(),
    &DDS_DYNAMIC_DATA_PROPERTY_DEFAULT);

// This call fails. If the above call used Foo_get_typecode instead then it would work

retcode = DDS_DynamicData_set_complex_member(data, ""my_typedef_struct"", 0, inner_data);
if (retcode != DDS_RETCODE_OK) {
    fprintf(stderr, ""_set_complex_member %d\n"", retcode);
    return -1;
}
```

But instead, it failed with these errors:

```
DDS_DynamicData2_copy: Objects have different types. self type = TypedefFoo, other type =
TypedefFoo
DDS_DynamicData2_finalize_ex: finalizing object bound to a member, automatically unbinding
now.
DDS_DynamicData2_set_complex_member:ERROR: Failed to copy value
DDS_DynamicData2_unbind_complex_member:ERROR: Bad parameter: self has no bound member
DDS_DynamicData2_set_complex_member:!unbind complex member
```

This issue has been fixed. Now, using either the aliased type (TypedefFoo in our example) or the original type (Foo in our example) works to set a complex member using the DynamicData API.

[RTI Issue ID CORE-12273]

5.8.9 Possible wrong results when adding Time or Duration objects that used very large numbers

Adding Time or Duration objects could have previously produced wrong results when using very large numbers. Necessary checks are now in place to ensure that wrong results do not occur.

[RTI Issue ID CORE-12413]

5.8.10 Java API did not support RtpsReliableReaderProtocol_t.receive_window_size

This QoS setting was ignored by the Java API, and readers were always created with the default value (256). This problem has been resolved.

[RTI Issue ID CORE-12451]

5.9 Fixes Related to Crashes

5.9.1 Simultaneous deletion of an entity by multiple threads caused a crash when using Java

When two threads deleted an entity at the same time, in Java, this may have caused a crash with the following backtrace:

```
#7 0x00007f7c630dad3b in REDAWeakReference_getReferent (reference=0x78,
slNode=0x7f7c4407f988, frOut=0x0, tableWithStartedCursor=0x7f7c6452c000) at
WeakReference.c:144
#8 0x00007f7c630d2ff3 in REDACursor_gotoWeakReference (c=0x7f7c4407f950, fr=0x0, wr=0x78) at
Cursor.c:230
#9 0x00007f7c62d5ed46 in PRESPsService_destroyLocalEndpoint (me=0x7f7c64367cc0,
failReason=0x7f7cb0136fc0, group=0x7f7c64dbb340, endpoint=0x7f7c644f0e88,
worker=0x7f7c44015f70) at PsService.c:2130
#10 0x00007f7c62b6fc26 in PRESParticipant_destroyLocalEndpoint (me=0x7f7c64368a00,
failReason=0x7f7cb0136fc0, group=0x7f7c64dbb340, endpoint=0x7f7c644f0e88,
worker=0x7f7c44015f70) at Participant.c:5882
#11 0x00007f7c636fcc32 in DDS_DataReader_deleteI (reader=0x7f7c644f1070) at
DataReader.c:4250
#12 0x00007f7c6372667e in DDS_Subscriber_delete_datareader (self=0x7f7c64dbb620,
reader=0x7f7c644f1070) at Subscriber.c:1159
#13 0x00007f7c63daf24b in Java_com_rti_dds_subscription_SubscriberImpl_DDS_1Subscriber_
1delete_1datareader (env=0x7f7c781061f8, self_class=0x7f7cb0137148, self=140172244792864,
readerL=140172235575408) at SubscriberImpl.c:790
```

This issue has been resolved. Now one thread will remove the entity and the other thread will throw an exception with the error code **com.rti.dds.infrastructure.RETCODE_ALREADY_DELETED**.

[RTI Issue ID CORE-10768]

5.9.2 DataReader C++ application crashed if it received tampered sample with unsupported encapsulation ID

If a C++ application with a *DataReader* received a sample with a tampered or malformed encapsulation kind, a segmentation fault occurred when the *DataReader* attempted to deserialize the sample, leading to an application crash. This problem has been fixed.

[RTI Issue ID CORE-12356]

5.9.3 Segmentation fault after calling DomainParticipant::register_durable_subscription with a group containing a long role_name

An application using the API **DomainParticipant::register_durable_subscription()** may have experienced a segmentation fault if the **role_name** of the input group was NULL or had a length greater than 512 bytes. This problem has been fixed.

[RTI Issue ID CORE-12460]

5.9.4 Segmentation fault when application using MultiChannel ran out of memory

A *Connext* application using MultiChannel might have produced a segmentation fault if the system ran out of memory. This problem has been fixed.

[RTI Issue ID CORE-12493]

5.9.5 Application crashed when capturing traffic for a DomainParticipant created before enabling network capture

To capture network traffic, you must enable this feature before creating the *DomainParticipants* that will capture the traffic. Applications not satisfying this requirement crashed when starting, pausing, or resuming the capture.

This problem has been fixed. *Connext* will no longer crash in this situation, but will fail and log messages such as the following:

```
ERROR NDDS_Utility_start_network_capture_w_params_for_participant:!get network capture
manager for DomainParticipant. Network capture must be enabled before creating the
DomainParticipant

ERROR NDDS_Utility_start_network_capture_for_participant:!network capture could not be
started for the participant

ERROR NDDS_Utility_run_network_capture_operation_for_all_participants:!failed to run network
capture operation for participant

ERROR NDDS_Utility_start_network_capture_w_params:!error starting network capture for all
participants

ERROR NDDS_Utility_start_network_capture:!start network capture for all participants. There
was at least one participant that could not be started
```

[RTI Issue ID CORE-12511]

5.9.6 Possible crash when writing a sample

Due to an internal error, an application could crash when writing a sample using either a best-effort or reliable *DataWriter*. Before the crash, an error message in either of the following functions was printed:

```
 * COMMENDBeWriterService_write
 * COMMENDSrWriterService_write
```

This problem has been resolved.

[RTI Issue ID CORE-12561]

5.9.7 Potential crash during type registration if system ran out of memory

A crash may have occurred during type registration if the application ran out of memory. This problem has been resolved.

[RTI Issue ID CORE-12734]

5.9.8 Segmentation fault after calling DomainParticipant::delete_durable_subscription with a group containing a long role_name

An application using the API **DomainParticipant::register_durable_subscription()** may have experienced a segmentation fault if the **role_name** of the input group was NULL or had a length greater than 512 bytes. This problem has been fixed.

[RTI Issue ID CORE-12787]

5.9.9 Potential crash or memory corruption if user application using thread-specific storage

Starting with release 6.1.0, there was an issue that could lead to a potential crash or memory corruption if the user application was using thread-specific storage.

In particular, when using Activity Context or Heap Monitoring, a race condition could have been triggered upon creating a thread with the ThreadFactory at the same time the DomainParticipantFactory instance was initialized or finalized. When this race condition was triggered, *Connext* might have overwritten the user application's thread-specific storage, leading to memory corruption or crashes.

This issue is now fixed. If the race condition that led to the issue happens in an application, the following benign warning will be logged:

```
Unexpected RTIOsapiContextSupport_g_tssKey value. This could mean that this thread was
created at the same time you are destroying the DDSDomainParticipantFactory.
```

If that is the case, Activity Context and Heap Monitoring won’t be available for that thread.

[RTI Issue ID CORE-12966]

5.10 Other Fixes

5.10.1 Serialization/deserialization of non-primitive sequences and arrays for XCDR2_DATA_REPRESENTATION did not follow Extensible Types specification

The serialization/deserialization of sequences and arrays with non-primitive members for XCDR2_DATA_REPRESENTATION did not follow the OMG 'Extensible and Dynamic Topic Types for DDS' specification, version 1.3. This led to compatibility issues with other DDS implementations.

This problem has been fixed, although the new behavior is not enabled by default, in order to keep backward compatibility with previous *Connext* releases. You can configure a *DomainParticipant* to align with the specification by setting **dds.type_plugin.dheader_in_non_primitive_collections** to true in the *DomainParticipant's* PROPERTY QoS Policy for all the *DomainParticipants* created by your *Connext* applications.

[RTI Issue ID CORE-12464]

5.10.2 Possible hang when using best-effort writers and asynchronous publishing

Due to an internal error, an application hung when using a best-effort writer and asynchronous publishing. Before the hang, the following error message was printed:

```
COMMENDBeWriterService_write:!retrieveJob
 This problem is now fixed.
```

[RTI Issue ID CORE-12562]

5.10.3 Unnecessary sockets created during initialization of library

The initialization of the *Connext* libraries always created a socket to obtain the IP address of the first valid interface of the machine. This IP address is used to generate identifiers when **DDS_DomainParticipantQos::wire_protocol::rtps_auto_id_kind** is **DDS_RTPS_AUTO_ID_FROM_IP**, which is not the default value. Therefore, the creation of this socket was unnecessary in most cases. This problem has been solved, and now the socket is only created when it is needed.

[RTI Issue ID CORE-12587]

5.10.4 Various issues with RtpsReliableWriterProtocol_t::nack_suppression_duration

There were various issues with the **RtpsReliableWriterProtocol_t::nack_suppression_duration** QoS:

- NACKs were being incorrectly suppressed with asynchronous publishing or non-zero **min/max_nack_response_delay** if two NACK messages were received within the **nack_suppression_duration** window, even if they were NACKing for different sets of sequence numbers. The **nack_suppression_duration** is only meant to suppress NACKs with the same leading sequence number that are received within the **nack_suppression_duration** window. If two consecutive NACKs have different leading sequence numbers, this indicates that the reader is making progress and the second one should not be suppressed, regardless of the **nack_suppression_duration**. Incorrect suppression of NACKs was not an issue if **min/max_nack_response_delay was zero and PublishModeQosPolicy.kind was SYNCHRONOUS_PUBLISH_MODE_QOS.**.
- If a NACK was received and suppressed due to the **nack_suppression_duration** before the previous NACK was responded to, then the NACK that had not been responded to yet, along with all NACKs for the duration of the **nack_suppression_duration**, were incorrectly suppressed. This problem did not occur if **min/max_nack_response_delay** was zero and **PublishModeQosPolicy.kind** was **SYNCHRONOUS_PUBLISH_MODE_QOS**.
- When **PublishModeQosPolicy.kind** was **ASYNCHRONOUS_PUBLISH_MODE_QOS**, if there were no GAP messages sent in response to a NACK, the **nack_suppression_duration** had no effect and NACKs were never suppressed. (GAP messages are sent to a *DataReader* to indicate that a sample or a set of samples are not meant for that *DataReader*. This can happen, for example, because the *DataWriter* has applied writer-side filtering or no longer has those samples in its queue.)

These issues have been resolved.

[RTI Issue ID CORE-12603]

5.10.5 Possible error message printed during Entity disposal

Upon the disposal of an entity, an error message from a callback associated with an event may have been printed. An excerpt of what the error may have looked like this:

```
ERROR [0x01013D3F,0x79453D76,0xA3558BB2:0x00000000|REMOVE REMOTE DR
0x01013D3F,0x79453D76,0xA3558BB2:0x80000007] OnReliableReaderActivityChangedCallback:An
exception was thrown: Omg.Dds.Core.DdsException: DDS operation failed:
  at Rti.Dds.NativeInterface.Helpers.ReturnCode.CheckResult(IntPtr result)
...
```

The disposal of entities has now been modified to ensure this error does not happen.

[RTI Issue ID CORE-12641]

5.10.6 Runtime error when using debug libraries for QNX x86 platform

When using the i86QNX6.6qcc_cpp4.7.3 debug libraries, your application may have had a runtime error and hung. This was because the debug libraries included the symbol for a math function (“**isinff**”) that was discontinued in QNX 6.3.

This problem has been resolved. The debug libraries now include "**isinf**" instead, which is supported.

A full list of the math functions that were discontinued in QNX 6.3 can be found here: http://www.qnx.-com/developers/docs/6.6.0.update/index.html#com.qnx.doc.neutrino.lib_ref/topic/whats_new_630.html.

Note: QNX platforms on x86 are not supported in Connext 7.0.0.

[RTI Issue ID CORE-12695]

5.10.7 Pushed samples may not have been received by reliable DataReader when DataWriter published Type that supports Zero Copy transfer over shared memory

A reliable *DataReader* may not have received pushed samples from a *DataWriter* publishing a *Topic* on a type configured with the zero-copy transfer over shared memory **@transfer_mode(SHMEM_ REF)**. This may have led to significant performance degradation because the *DataReader* has to continuously NACK the missing samples.

This problem only occurred when the following three conditions were true:

1. The *DataWriter* ran in a different host, or the *DataReader* did not have the builtin SHMEM transport enabled.
2. The *DataReader* used a ContentFilteredTopic, and the *DataWriter* did writer-side filtering, or the *DataReader* created TopicQueries.
3. The *DataWriter* was not configured to use an asynchronous publisher. This problem has been resolved.

[RTI Issue ID CORE-12775]

5.10.8 Unbounded memory growth in Monitoring Library when creating and deleting endpoints

Each time a *DataWriter* or *DataReader* is created in an application that has the RTI Monitoring Library enabled, a new instance is created in the *DataWriters* of the library. Since, by default, the maximum number of instances the *DataWriter* can handle is unlimited, and the instances of already deleted endpoints were not cleaned up automatically, unbounded memory growth was possible in the library's *DataWriters* when constantly creating and deleting endpoints in an application that had Monitoring Library enabled.

This problem has been fixed by setting the **writer_data_lifecycle::autopurge_disposed_instances_ delay** QoS to **DDS_DURATION_ZERO** by default in the *DataWriters* for the Monitoring Library. That way, disposed instances will be instantly cleared.

[RTI Issue ID MONITOR-244]

5.10.9 Unexpected behavior when two threads crashed at the same time on Windows systems

When two threads crashed at the same time on Windows systems, *Connext* may have concurrently called the function **SymInitialize()** from **DbgHelp** from two crashing threads.

SymInitialize() is not thread safe, so the application may have run into unexpected behavior or memory corruption under this scenario.

This has been resolved, *Connext* no longer calls **SymInitialize()** from a crashing thread. Instead, **SymInitialize()** is now called during DomainParticipantFactory initialization.

[RTI Issue ID CORE-10066]

5.10.10 DataReaders setting reader_qos.protocol.expects_inline_qos to TRUE incorrectly matched with DataWriters

Connext DataWriters matched *DataReaders* that set **reader_qos.protocol.expects_inline_qos** to TRUE. This behavior was incorrect because *Connext DataWriters* do not support sending inline QoS, and they were not honoring the *DataReaders*' requests.

This issue has been fixed. The behavior has changed so that *DataWriters* will not match *DataReaders* that request inline QoS (i.e., that set **reader_qos.protocol.expects_inline_qos** to TRUE).

[RTI Issue ID CORE-10501]

5.10.11 Source IP on Spy was not correct when DataWriters with same Topic were on different machines

The source IP on Spy may not have been correct when *DataWriters* with the same Topic were on different machines. This issue has been fixed. Now the source IP is per Entity, not per Topic, and the output will look like this:

```
11:35:13 New reader from 10.200.130.20 : topic=""Example app"" type=""app""
11:35:18 New writer from 10.200.129.195 : topic=""Example app"" type=""app""
11:35:16 New writer from 10.200.130.3 : topic=""Example app"" type=""app""
11:42:58 New data from 10.200.129.195 : topic=""Example app"" type=""app""
11:42:58 New data from 10.200.130.3 : topic=""Example app"" type=""app""
11:43:00 New data from 10.200.129.195 : topic=""Example app"" type=""app""
11:43:00 New data from 10.200.130.3 : topic=""Example app"" type=""app""
```

[RTI Issue ID CORE-12169]

5.10.12 Writer using durable writer history may not have blocked after send window filled up when disable positive ACKs was enabled

In previous releases, a reliable *DataWriter* configuring a finite send window size may not have blocked when the send window filled up if all these conditions were met:

- *DataWriter* was configured to use durable writer history.
- *DataWriter* was configured to use disable positive ACKs.
- *DataWriter* was configured with **writer_qos.reliability.acknowledgment_kind** set to AUTO or EXPLICIT, or **writer_qos.availability.enable_required_subscriptions** was set to TRUE.

Because of this issue, the reliability protocol for the *DataWriter* may have been less efficient. This problem has been resolved.

[RTI Issue ID CORE-12225]

5.10.13 Potential truncation of application-level acknowledgment response data

Connext enforced a wrong maximum length for application-level acknowledgment response data. Specifically, *Connext* incorrectly allowed setting the DATA_READER_RESOURCE_LIMITS QosPolicy's **max_app_ack_response_length** longer than the maximum serializable data, which resulted in the truncation of data when the length got close to 64kB.

This problem has been resolved: *Connext* now enforces a maximum length of 32kB for **max_app_ack_response_length** as part of *DataReader* QoS consistency checks, and it will log an error if you try to set **max_app_ack_response_length** longer than 32kB.

[RTI Issue ID CORE-12450]

5.10.14 Error messages displayed that should not have been, when printing DataReaderQoS objects

When printing DataReaderQoS objects, and the contained DDSOwnershipQosPolicy or DDS_TransportMulticastQosPolicy policies were printed, some error messages were displayed that should not have been. These error messages could have been safely ignored by an application. These error messages are no longer printed.

[RTI Issue ID CORE-12462]

5.10.15 Potential Valgrind invalid read when logging a message or enabling heap monitoring

When activity context was enabled in logging, or when heap monitoring was enabled, a Valgrind invalid read similar to the following one may have been reported:

```
==1344490== Invalid read of size 4
==1344490== at 0x4A3FA0A: RTIOsapiActivityContext_skipResourceGuid (ActivityContext.c:246)
==1344490== by 0x4A417B3: RTIOsapiActivityContext_getString (ActivityContext.c:820)
```

This issue has been resolved. The Valgrind invalid read error no longer appears.

[RTI Issue ID CORE-12537]

5.10.16 Malformed IDL printed if multiple labels used for default case of a union

The IDL produced by the C API's **DDS_TypeCode_print_IDL()** function (or the equivalent in other APIs) may have been malformed if multiple labels were assigned to the default case of a union. All of the labels were printed as "default: ", instead of their true value. This problem has been resolved.

[RTI Issue ID CORE-12624]

Chapter 6 Known Issues

Note: For an updated list of critical known issues, see the Critical Issues List on the RTI Customer Portal at https://support.rti.com.

6.1 Known Issues with Usability

6.1.1 Cannot open USER_QOS_PROFILES.xml in rti_workspace/examples from Visual Studio

When trying to open the **USER_QOS_PROFILES.xml** file from the resource folder of one of the provided examples, you may see the following error:

```
Could not find file : C:\Users\<user>\Documents\rti_workspace\5.3.0\examples\connext_
dds\c\<example>\win32\USER_QOS_PROFILES.xml
```

The problem is that the Visual Studio project is looking for the file in a wrong location (win32 folder).

You can open the file manually from here:

C:\Users\<user>\Documents\rti_workspace\5.3.0\examples\connext_dds\c\<example>\USER_QOS_PROFILES.xml

This issue does not affect the functionality of the example.

[RTI Issue ID CODEGENII-743]

6.1.2 DataWriter's Listener callback on_application_acknowledgment() not triggered by late-joining DataReaders

The *DataWriter's* listener callback **on_application_acknowledgment()** may not be triggered by late-joining *DataReaders* for a sample after the sample has been application-level acknowledged by all live *DataReaders* (no late-joiners).

If your application requires acknowledgment of message receipt by late-joiners, use the Request/Reply communication pattern with an Acknowledgment type (see the chapter "

Introduction to the Request-Reply Communication Pattern," in the *RTI Connext Core Libraries User's Manual*).

[RTI Issue ID CORE-5181]

6.1.3 HighThroughput and AutoTuning built-in QoS Profiles may cause communication failure when writing small samples

If you inherit from either the **BuiltinQosLibExp::Generic.StrictReliable.HighThroughput** or the **BuiltinQosLibExp::Generic.AutoTuning** built-in QoS profiles, your *DataWriters* and *DataReaders* will fail to communicate if you are writing small samples.

In *Connext* 5.1.0, if you wrote samples that were smaller than 384 bytes, you would run into this problem. In version 5.2.0 onward, you might experience this problem when writing samples that are smaller than 120 bytes.

This communication failure is due to an interaction between the batching QoS settings in the **Generic.HighThroughput** profile and the *DataReader*'s **max_samples** resource limit, set in the **BuiltinQosLibExp::Generic.StrictReliable** profile. The size of the batches that the *DataWriter* writes are limited to 30,720 bytes (see max_data_bytes). This means that if you are writing samples that are smaller than 30,720/**max_samples** bytes, each batch will have more than **max_samples** samples in it. The *DataReader* cannot handle a batch with more than **max_samples** samples and the batch will be dropped.

There are a number of ways to fix this problem, the most straightforward of which is to overwrite the *DataReader's***max_samples** resource limit. In your own QoS profile, use a higher value that accommodates the number of samples that will be sent in each batch. (Simply divide 30,720 by the size of your samples).

[RTI Issue ID CORE-6411]

6.1.4 Memory leak if Foo:initialize() called twice

Calling **Foo:initialize()** more than once will cause a memory leak.

[RTI Issue ID CORE-7678]

6.1.5 Wrong error code after timeout on write() from Asynchronous Publisher

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

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

6.1.6 Type Consistency enforcement disabled for structs with more than 10000 members

TypeObjects cannot be created from structs with more than 10000 members. Applications that publish or subscribe to such types may see errors like the following:

```
RTICdrStream_serializeNonPrimitiveSequence:sequence length (10005) exceeds maximum (10000)
RTICdrTypeObjectTypeLibraryElement_getTypeId:serialization error: Type
RTICdrTypeObject_fillType:!get TypeId
RTICdrTypeObject_assertTypeFromTypeCode:!create Structure Type
RTICdrTypeObject_createFromTypeCode:!create TypeObject
```

When the TypeObject can't be serialized, the type compatibility check between a reader and a writer falls back to exact type-name matching.

See the section "Verifying Type Consistency: Type Assignability" in the *RTI Connext Core Libraries Extensible Types Guide* for more information.

[RTI Issue ID CORE-8158]

6.1.7 Escaping special characters in regular/filter expressions not supported in some cases

Escaping special characters is not supported in expressions when using the following features:

- Partitions
- MultiChannel

Every occurrence of a backslash ('\') will be considered its own character and not a way to escape the character that follows. For example: 'A\?' does not match 'A?' because the first expression is considered an expression with three characters.

[RTI Issue ID CORE-11858]

6.2 Known Issues with Code Generation

6.2.1 Examples and generated code for Visual Studio 2017 and later may not compile (Error MSB8036)

The examples provided with *Connext* and the code generated for Visual Studio 2017 and later will not compile out of the box if the Windows SDK version installed is not a specific number like 10.0.15063.0. If that happens, you will see the compilation error MSB8036. To compile these projects, select an installed version of Windows SDK from the Project menu -> Retarget solution.

Another option is to set the enviroment variable RTI_VS_WINDOWS_TARGET_PLATFORM_VERSION to the SDK version number. For example, set RTI_VS_WINDOWS_TARGET_

PLATFORM_VERSION to 10.0.16299.0. (Note: the environment variable will not work if you have already retargeted the project via the Project menu.)

For further details, see the Windows chapter of the *RTI Connext Core Libraries Platform Notes*.

[RTI Issue ID CODEGENII-800]

6.3 Known Issues with Instance Lifecycle

6.3.1 Instance does not transition to ALIVE when "live" DataWriter detected

The "Data Distribution Service for Real-time Systems" specification allows transitioning an instance from the NO_WRITERS state to the ALIVE state when a "live" *DataWriter* writing the instance is detected. Currently, this state transition is not supported in *Connext*. The only way to transition an instance from NO_WRITERS to ALIVE state is by receiving a sample for the instance from one of the *DataWriters* publishing it.

Example:

1. A *DataWriter* writes a particular instance. The *DataReader* receives the sample. The *DataWriter* loses liveliness with the *DataReader*, making the instance transition from ALIVE to NO_WRITERS. The writer later becomes alive again, but it doesn't resume writing samples of the instance. In this case, the instance will stay in a NO_WRITERS state.
2. The *DataWriter* publishes a new sample for the instance. Only then does the instance state change on the *DataReader* from NO_WRITERS to ALIVE.

Note: To address this known issue, in 7.1.0, RTI has introduced a new QoS parameter, **instance_state_recovery_kind**, with the experimental option RECOVER_INSTANCE_STATE_RECOVERY, in the RELIABILITY QosPolicy. Because it is an experimental feature, please do not use the new option in deployed systems. For information, see the "Transition after NOT_ALIVE_NO_WRITERS" section in *Instance States*, in the RTI Connext Core Libraries User's Manual.

[RTI Issue ID CORE-3018]

6.3.2 Persistence Service DataReaders ignore serialized key propagated with dispose updates

Persistence Service DataReaders ignore the serialized key propagated with dispose updates. *Persistence Service DataWriters* cannot propagate the serialized key with dispose, and therefore ignore the **serialize_key_with_dispose** setting on the *DataWriter* QoS.

[RTI Issue ID PERSISTENCE-221]

6.4 Known Issues with Reliability

6.4.1 DataReaders with different reliability kinds under Subscriber with GROUP_PRESENTATION_QOS may cause communication failure

Creating a *Subscriber* with **PresentationQosPolicy.access_scope** GROUP_PRESENTATION_QOS and then creating *DataReaders* with different **ReliabilityQosPolicy.kind** values creates the potential for situations in which those *DataReaders* will not receive any data.

One such situation is when the *DataReaders* are discovered as late-joiners. In this case, samples are never delivered to the *DataReaders*. A workaround for this issue is to set the **Avail-abilityQosPolicy.max_data_availabilty_waiting_time** to a finite value for each *DataReader*.

[RTI Issue ID CORE-7284]

6.5 Known Issues with Content Filters and Query Conditions

6.5.1 Writer-side filtering may cause missed deadline

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

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

6.5.2 filter_sample_* statistics in DDS_DataWriterProtocolStatus not updated correctly

The **filter_sample_*** statistics in the **DDS_DataWriterProtocolStatus** are not updated correctly. The values that you get after calling the following APIs may be smaller than the actual values:

- **DDS_DataWriter::get_datawriter_protocol_status**
- **DDS_DataWriter::get_matched_subscription_datawriter_protocol_status**
- **DDS_DataWriter::get_matched_subscription_datawriter_protocol_status_by_locator**

[RTI Issue ID CORE-5157]

6.6 Known Issues with TopicQueries

6.6.1 TopicQueries not supported with DataWriters configured to use batching or Durable Writer History

Getting TopicQuery data from a *DataWriter* configured to use batching or Durable Writer History is not supported.

[RTI Issue IDs CORE-7405, CORE-7406]

6.7 Known Issues with Transports

6.7.1 AppAck messages cannot be greater than underlying transport message size

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

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

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

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

Why does an AppAck message increase its size? An AppAck message contains a list of sequence number intervals where each interval represents a set of consecutive sequence numbers that have been already acknowledged. As long as samples are acknowledged in order, the AppAck message will always have a single interval. However, when samples are acknowledged out of order, the number of intervals and the size of the AppAck will increase.

For more information, see the "Application Acknowledgment" section in the *RTI Connext Core Libraries User's Manual*.

[RTI Issue ID CORE-5329]

6.7.2 DataReader cannot persist AppAck messages greater than 32767 bytes

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

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

For more information, see the section "Durable Reader State," in the *RTI Connext Core Libraries User's Manual*.

[RTI Issue ID CORE-5360]

6.7.3 Discovery with Connext Micro fails when shared memory transport enabled

Given a *Connext* application with the shared memory transport enabled, a *Connext* Micro 2.4.x application will fail to discover it. This is due to a bug in *Connext* Micro that prevents a received participant discovery message from being correctly processed. This bug will be fixed in a future release of *Connext* Micro. As a workaround, you can disable the shared memory transport in the *Connext* application and use UDPv4 instead.

[RTI Issue ID EDDY-1615]

6.7.4 Communication may not be reestablished in some IP mobility scenarios

If you have two *Connext* applications in different nodes and they change their IP address at the same time, they may not reestablish communication. This situation may happen in the following scenario:

- The applications see each other only from one single network.
- The IP address change happens at the same time in the network interface cards (NICs) that are in the network that is in common for both applications.
- The IP address change on one of the nodes happens before the arrival of the DDS discovery message propagating the address change from the other side.

[RTI Issue ID CORE-8260]

6.7.5 Corrupted samples may be forwarded through Routing Service when using Zero-Copy transfer over shared memory

When using *Zero Copy transfer over shared memory* together with *RTI Routing Service*, *Routing Service* avoids an additional copy of the data by passing a reference to the sample from the input to the output of a route. If the sample is reused and rewritten by the original application *DataWriter* during the time between when the sample was received on the route input and copied into the route output buffer, the forwarded sample will contain the updated, and now invalid, values for the original sample.

This situation can be avoided in a few different ways, with various tradeoffs.

6.7.5.1 Use automatic application acknowledgment

Using automatic application acknowledgment (**acknowledgment_mode** = APPLICATION_AUTO_ACKNOWLEDGMENT in the Reliability QoS Policy) between the *Routing Service* input *DataReader* and its matching *DataWriters* will avoid the issue.

When using Zero Copy transfer over shared memory, *DataWriters* must loan samples using the **get_loan** API. Only samples that have been fully acknowledged will be returned by the **get_loan** API. This means that if automatic application acknowledgment is turned on, that only samples that the *Routing*

Service has already copied and written to the route output will be available for reuse by the original *DataWriter*, because *Routing Service* does not return the loan on a sample until after it is forwarded to the route outputs.

The drawback to this approach is that it requires RELIABLE Reliability. In addition, application-level acknowledgments are not supported in *Connext Micro*, so this approach will not work if *Connext Micro* is the source of the Zero Copy samples.

6.7.5.2 Ensure that the number of available samples accounts for Routing Service processing time

Regardless of whether you are using *Routing Service*, it is important when using Zero Copy transfer over shared memory to size your resources so that your application can continue to write at the desired rate while the receiving applications receive and process the samples. If you are using *Routing Service* and cannot, or do not wish to, use automatic application acknowledgments, you must take into account the amount of time it will take to receive and forward a sample when setting **writer_loaned_sample_allocation** in the DATA_WRITER_RESOURCE_LIMITS QoS Policy and managing the samples in your application.

[RTI Issue ID CORE-10782]

6.7.6 Network Capture does not support frames larger than 65535 bytes

Network capture does not support frames larger than 65535 bytes. This limitation affects the TCP transport protocol if the **message_size_max** property is set to a value larger than the default one.

[RTI Issue ID CORE-11083]

6.8 Known Issues with FlatData

6.8.1 FlatData language bindings do not support automatic initialization of arrays of primitive values to non-zero default values

RTI FlatData™ language bindings do not support the automatic initialization of arrays of primitive values to non-zero default values, unless the primitive is an enumeration. It is possible to declare an alias to a primitive member with a default value using the @default annotation, and then to declare an array of that alias. For example:

```
@default(10)
typedef int32 myLongAlias;

struct MyType {
    myLongAlias myLongArray[25];
};
```

The default values of each member of the array in this case should be 10, but in FlatData they will all be set to 0.

[RTI Issue ID CORE-9176]

6.8.2 Flat Data: plain_cast on types with 64-bit integers may cause undefined behavior

The function **rti::flat::plain_cast** is allowed on FlatData samples containing int64_t members, but those members are not guaranteed to have an 8-byte alignment (a 4-byte alignment is guaranteed). Memory checkers such as Valgrind may report errors when accessing such members from the pointer returned by **plain_cast**.

[RTI Issue ID CORE-10092]

6.8.3 FlatData in combination with payload encryption in RTI Security Plugins and/or compression will not save copies

RTI FlatData™ language binding offers a reduced number of end-to-end copies when sending a sample (from four to two), providing improved latency for large data samples. (See the "FlatData Language Binding" section in the *RTI Connext Core Libraries User's Manual*.) When used with payload encryption and/or payload compression, however, there are no savings in the number of copies. (See the section "Interactions with *RTI Security Plugins* and Compression" in the "Using FlatData Language Binding" section of the *RTI Connext Core Libraries User's Manual*). In future releases, other copies currently being made can potentially be optimized out in order to reduce the number of copies when using FlatData in combination with security and compression.

[RTI Issue ID CORE-11262]

6.9 Known Issues with Coherent Sets

6.9.1 Some coherent sets may be lost or reported as incomplete with batching configurations

If *Connext* 6.1.0 receives coherent sets from *Connext* 6.0.0 or lower using batching, coherent sets that are fully received and complete may be lost or marked as incomplete. (If the QoS **subscriber_qos.-presentation.drop_incomplete_coherent_set** is set to FALSE, then the samples marked as incomplete won't be dropped.)

[RTI Issue ID CORE-9691]

6.9.2 Copy of SampleInfo::coherent_set_info field is not supported

SampleInfo::coherent_set_info is not available when using take/read operations that do not loan the samples. The **SampleInfo::coherent_set_info** is always set to NULL when you call the take/read operations that do not loan the samples. To get the **coherent_set_info** value, make sure you use the read/-take operations that loan the data.

In addition, the copy constructor and assignment operator in the Traditional C++ and Modern C++ APIs do not copy the **SampleInfo::coherent_set_info** field. It is always set to NULL. It is your responsibility to make the copy and handle memory allocation and deletion for this field.

[RTI Issue ID CORE-11215]

6.9.3 Other known issues with coherent sets

Coherent sets are not propagated through *RTI Routing Service* [RTI Issue ID ROUTING-657].

Group coherent sets are not supported with ODBC writer history [RTI Issue ID CORE-9746].

Group coherent sets are not persisted by *RTI Persistence Service* [RTI Issue ID PERSISTENCE-191].

Group coherent sets cannot be stored or replayed with *RTI Recording Service* [RTI Issue ID RECORD-1083].

6.10 Known Issues with Dynamic Data

6.10.1 Conversion of data by member-access primitives limited when converting to types that are not supported on all platforms

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 int64 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 int64s from a 32-bit or smaller integer type is supported on all Windows and Linux architectures, and any additional 64-bit architectures. Conversion to 128-bit long doubles from a float or double is not supported.

[RTI Issue ID CORE-2986]

6.10.2 Types that contain bit fields not supported

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

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

[RTI Issue ID CORE-3949]

6.11 Known Issues in RTI Monitoring Library

6.11.1 Problems with NDDS_Transport_Support_set_builtin_transport_property () if Participant Sends Monitoring Data

If a *Connext* application uses the **NDDS_Transport_Support_set_builtin_transport_property()** API (instead of the PropertyQosPolicy) to set built-in transport properties, it will not work with *Monitoring Library* if the user participant is used for sending all the monitoring data (the default settings). As a workaround, you can configure Monitoring Library to use another participant to publish monitoring data (using the property name **rti.monitor.config.new_participant_domain_id** in the PropertyQosPolicy).

[RTI Issue ID MONITOR-222]

6.11.2 Participant's CPU and memory statistics are per application

The CPU and memory usage statistics published in the *DomainParticipant* entity statistics topic are per application instead of per *DomainParticipant*.

[RTI Issue ID CORE-7972]

6.11.3 XML-based entity creation nominally incompatible with static monitoring library

If setting the *DomainParticipant* QoS programmatically in the application is not possible (i.e., when using XML-based Application Creation), the monitoring **create** function pointer may still be provided via an XML profile by using the environment variable expansion functionality. The monitoring property within the *DomainParticipant* QoS profile in XML must be set as follows:

```
<domain_participant_qos>
    <property>
        <value>
            <element>
                <name>rti.monitor.library</name>
                <value>timonitoring</value>
            </element>
            <element>
                <name>rti.monitor.create_function_ptr</name>
                <value>$(MONITORFUNC)</value>
            </element>
        </value>
    </property>
</domain_participant_qos>
```

Then in the application, before retrieving the DomainParticipantFactory, the environment variable must be set programmatically as follows:

```
...
sprintf(varString, "MONITORFUNC=%p", RTIDefaultMonitor_create);
int retVal = putenv(varString);
...
//DomainParticipantFactory must be created after env. variable setting
```

[RTI Issue ID CORE-5540]

6.11.4 ResourceLimit channel_seq_max_length must not be changed

The default value of **DDS_DomainParticipantResourceLimitsQosPolicy::channel_seq_max_length** can't be modified if a *DomainParticipant* is being monitored. If this QoS value is modified from its default value of 32, the monitoring library will fail.

[RTI Issue ID MONITOR-220]

6.12 Known Issues with Installers

6.12.1 RTI Connext Micro 3.0.3 installation package currently compatible only with Connext 6.0.1 installer

Connext Micro 3.0.3 must be installed with *Connext Professional* release 6.0.1. Note that, although you cannot install *Connext Micro* 3.0.3 with *Connext* 6.1.0 and higher, it is interoperable with *Connext* 6.1.0 and higher, except as noted in the *Migration Guide* on the RTI Community Portal (https://-community.rti.com/documentation).

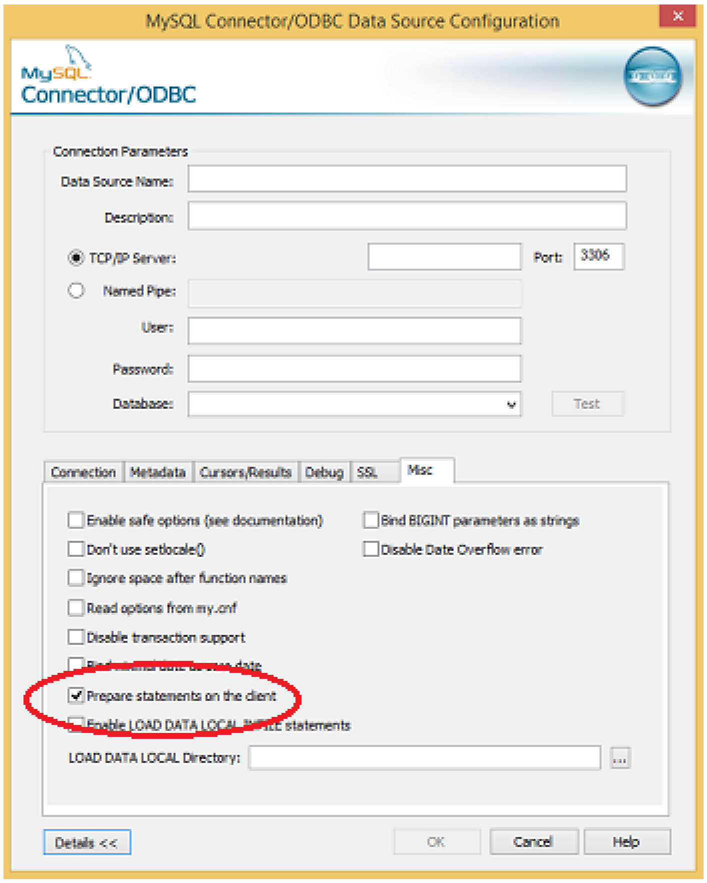
6.13 Other Known Issues

6.13.1 Possible Valgrind still-reachable leaks when loading dynamic libraries

If you load any dynamic libraries, you may see "still reachable" memory leaks in "dlopen" and "dlclose". These leaks are a result of a bug in Valgrind (https://bug-s.launchpad.net/ubuntu/+source/valgrind/+bug/1160352).

This issue affects the *Core Libraries*, *Security Plugins*, and *TLS Support*.

[RTI Issue IDs CORE-9941, SEC-1026, and COREPLG-510]



Connection Parameters
Data Source Name:
Description:
(●) TCP/IP Server: Port: 3306
( ) Named Pipe:
User:
Password:
Database:

[ ] Enable safe options (see documentation)
[ ] Don't use setlocale()
[ ] Ignore space after function names
[ ] Read options from my.cnf
[ ] Disable transaction support
[ ] Bind minimal date as zero date
[✓] Prepare statements on the client
[ ] Enable LOAD DATA LOCAL INFILE statements
LOAD DATA LOCAL Directory:
[ ] Bind BIGINT parameters as strings
[ ] Disable Date Overflow error

6.13.2 64-bit discriminator values greater than (2^31-1) or smaller than (-2^31) supported only in Java, no other languages

Unions with a 64-bit integer discriminator type containing discriminator values that cannot fit in a 32-bit value are not supported when using the following language bindings:

- C
- Traditional C++
- Modern C++
- New. NET
- DynamicData (regardless of the language)

They are also not supported with ContentFilteredTopics, regardless of the language binding.

Using label values greater than 32-bit may lead to receiving samples with invalid content or to filtering samples incorrectly.

For example, this is not supported:

```
union union_uint64 switch (uint64) {
    case 0x100000000:
              char m_char;
    case 0x200000000:
              int32 m_int32;
    case 0x300000000:
              string<5> m_string;
};
```

This is supported:

```
union union_uint64 switch (uint64) {
    case 1:
              char m_char;
    case 2:
              int32 m_int32;
    case 3:
              string<5> m_string;
};
```

[RTI Issue ID CORE-11437]

6.13.3 Creating multiple DataReaders for the same Topic under the same Subscriber configured with Group Ordered Access is not supported

Creating multiple *DataReaders* for the same *Topic* under the same *Subscriber* configured with PresentationQosPolicy **access_scope** = GROUP and **ordered_access** = TRUE is not supported. If you try to create a second reader in this situation, it will fail to be created and this error will be printed:

```
ERROR [0x0101E967,0x5C3A43B1,0x99D71EB7:0x80000309{Entity=Su,Domain=0}|CREATE DR WITH TOPIC
FooTopic|LC:DISC]PRESPsService_createLocalEndpoint:NOT SUPPORTED | Creating more than one
reader for the same topic within a single subscriber using GROUP presentation and ordered_
access=true.
```

Instead, in this situation, you will need to use only one *DataReader*, or you will need to create a new *Subscriber* and *DataReader* in the same *DomainParticipant*.

[RTI Issue ID CORE-12448]

Chapter 7 Experimental Features

This software may contain experimental features. These 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.

In the API Reference HTML documentation, experimental APIs are marked with **<<experimental>>**.

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);
```

Experimental features are also clearly noted as such in the *User's Manual* or *Getting Started Guide* for the component in which they are included.

Disclaimers:

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

Please submit your comments and suggestions about experimental features to **support@rti.com** or via the RTI Customer Portal (https://support.rti.com/). Although the RTI Support team does not provide support for experimental features, you may be able to get help with experimental features from the RTI Community forum: https://community.rti.com/.

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/