RTI Connext DDS

Core Libraries

Release Notes

Version 6.1.1

Printed in U.S.A. First printing.
June 2022.

Trademarks

Copy and Use Restrictions

Notice

Any deprecations noted in this document serve as notice under the Real-Time Innovations, Inc. Maintenance Policy #4220.

Technical Support

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

Contents

Chapter 1 Introduction

RTI® Connext® DDS 6.1.1 LTS is a long-term support release. See the Connext LTS page on the RTI website for more information about long-term support releases.

This document includes the following:

For an overview of new features in 6.1.1, see RTI Connext DDS Core Libraries What's New in 6.1.1.

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 DDS*, 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 Supported Operating Systems

Connext DDS 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 DDS* 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 DDS* application for any architecture. You will also need a target installation, which provides the libraries required to build a *Connext DDS* application for that particular target architecture.

Connext DDS is available for the platforms in the following table.

Table 2.1 Supported Platforms

Operating System	Version and CPU
Android™ (target only)	Android 9.0 on Arm v7 and v8 (available on demand)
INTEGRITY® (target only)	INTEGRITY 11.0.4 on p4080, 11.4.4 on x64
Linux® on Arm® CPUs (target only)	NI™ Linux 3 on Arm v7 Ubuntu® 18.04 LTS on Arm v7 and Arm v8 Ubuntu 16.04 LTS on Arm v8 (available on demand)

Table 2.1 Supported Platforms

Operating System	Version and CPU
Linux on Intel® CPUs (host and target)	CentOS™ 6.0, 6.2-6.4 on x64 (available on demand) CentOS 7.0 on x64 Red Hat® Enterprise Linux 6.0-6.5, 6.7, 6.8 on x64 (available on demand) Red Hat Enterprise Linux 7.0, 7.3, 7.5, 7.6, 8.0 on x64 SUSE® Linux Enterprise Server 12 SP2 on x64 Ubuntu 16.04 LTS on x64 (available on demand) Ubuntu 18.04 LTS, 20.04 LTS on x64 POSIX-compliant platforms, made available with *RTI Connext TSS:* CentOS 7.0 Red Hat Enterprise Linux 7, 7.3, 7.5, 7.6, 8 Ubuntu 18.04 LTS
macOS on Arm CPUs (target only)	macOS 11 on Arm v8 (*Requires Rosetta® 2 during installation, not required during runtime.)*
macOS® on Intel CPUs (host and target)	macOS 10.13 - 10.15 on x64 macOS 11 on x64
QNX® (target only)	QNX Neutrino® 6.4.1 on x86 QNX Neutrino 6.5 on x86 QNX Neutrino 6.5 SP1 on Arm v7 QNX Neutrino 7.0.4 on x64 and Arm v8 QNX Neutrino 7.1 on Arm v8
VxWorks® (target only)	VxWorks 6.9.3.2 on x64 VxWorks 6.9.4.2 and 6.9.4.6 on PPC VxWorks 7.0 SR0510 and SR0630 on x64
Windows® (host and target)	Windows 10 on x64 [a] Windows Server 2016 on x64

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

The following table lists additional target libraries for which RTI offers custom support. If you are interested in using one of these platforms, please contact your local RTI sales representative or email **sales@rti.com**.

[a] Per Microsoft, this should be compatible with Windows 10 IoT Enterprise with Windows native app.

Table 2.2 Custom Supported Platforms

Operating System	Version and CPU
AIX® (host and target)	AIX 7.2 on POWER9™
INtime® (target only)	INtime 6.3 on 64-bit Windows 10 on x86 (available on demand)
Linux® on Intel CPUs (target only)	RedHawk™ Linux 6.0 on x64 (available on demand)
	RedHawk Linux 6.5 on x86 and x64 (available on demand)
Linux on Arm CPUs (target only)	Wind River® Linux 8 on Arm v7
	Yocto Project® 2.5 on Arm v8
QNX® (target only)	QNX Neutrino 6.5 on PPC e500v2
	QNX Neutrino 6.6 on Arm v7 and x86
	QNX Neutrino 7.0.4 on Arm v7
	QNX Neutrino 7.1 on 64-bit Arm v8, compatible with wolfSSL 4.7

2.2 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 2012 — Update 4 Redistributable Package Requirement

You must have the Visual C++ Redistributable for Visual Studio 2012 Update 4 installed on the machine where you are *running* an application linked with dynamic libraries. This includes dynamically linked C/C++ and all .NET and Java applications.

You can download Visual C++ Redistributable for Visual Studio 2012 Update 4 from this Microsoft web-site: http://www.microsoft.com/en-ca/download/details.aspx?id=30679

When Using Visual Studio 2013 — Redistributable Package Requirement

You must have Visual C++ Redistributable for Visual Studio 2013 installed on the machine where you are *running* an application linked with dynamic libraries. This includes C/C++ dynamically linked and all .NET and Java applications.

You can download Visual C++ Redistributable for Visual Studio 2013 from this Microsoft website: https://www.microsoft.com/en-us/download/details.aspx?id=40784

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 .NET and 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 .NET and 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 .NET and 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".

2.3 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 6.1.1 and previous releases, see the *Migration Guide* on the RTI Community Portal (https://community.rti.com/documentation).

3.1 Wire Protocol Compatibility

Connext DDS 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 2.3. RTI plans to maintain interoperability between middleware versions based on RTPS 2.1. For more details, see the "RTPS Versions" section of the *Migration Guide* on the RTI Community Portal (https://-community.rti.com/documentation).

3.2 Code and Configuration Compatibility

The *Connext DDS* 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 DDS* 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 DDS* 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 DDS Core Libraries Extensible Types Guide* for a full list of the supported and unsupported extensible types features.

3.4 ODBC Database Compatibility

To use the Durable Writer History and Durable Reader State features, you must install a relational database such as MySQL.

To see if a specific architecture has been tested with the Durable Writer History and Durable Reader State features, see the *RTI Connext DDS Core Libraries Platform Notes*. To see what databases are supported, see *RTI Connext DDS Core Libraries Database Setup*.

Chapter 4 What's Fixed in 6.1.1

This section describes bugs fixed in *Connext DDS* 6.1.1. These fixes have been made since 6.1.0 was released.

4.1 Fixes Related to Discovery

4.1.1 Large memory allocation on DataReaders due to tampered discovery messages

Corrupted or tampered samples could entail large memory allocations up to a maximum of 2Gb of memory dynamically allocated. This was an issue because, even if security could be used to protect some channels, there were others that discovery depends upon that could not be protected.

This issue has been fixed. Now, corrupted or tampered samples do not entail large memory allocations.

[RTI Issue ID CORE-11319]

4.2 Fixes Related to Serialization and Deserialization

4.2.1 Serialization of string members did not check for null-terminated strings in C, traditional C++, and modern C++

The code executed by a *DataWriter* that serializes string members in a Topic type did not check that the strings were null-terminated. This may have led to undefined behavior, because the serialization code calls **strlen**.

This problem has been fixed. The serialization code now checks for null-terminated strings with the maximum allowed length and reports the following error if the string is not well-formed:

```
RTIXCdrInterpreter_serializeString:StrStruct:member2 serialization error. String length
(at least 6) is larger than maximum 5
```

[RTI Issue ID CORE-11164]

4.2.2 Error serializing samples containing strings with NULL character in the middle, in modern C++

Samples containing string members with a NULL character in the middle were not serialized correctly. This may have led to subscribing applications not receiving the sample data.

This issue only affected modern C++, where **std::string** and **std::wstring** can have a NULL character in the middle. With the fix, the string is now truncated up to the first NULL character.

[RTI Issue ID CORE-11308]

4.2.3 Invalid key deserialization for mutable derived types with key members

In 6.1.0 and 6.0.1.x releases, the key deserialization for mutable derived types with key members when the base does not contain keys was invalid. For example:

```
@mutable
struct Base1 {
    long m1;
};

@mutable
struct Derived1 : Base1 {
    @key long m2;
};
```

This issue affected the following functionality:

- Calling the APIs **DataWriter::get_key_value** and **DataReader::get_key_value** returned an invalid value.
- When **writer_qos.protocol.serialize_key_with_dispose** was set to TRUE (not the default value) and **writer_qos.protocol.disable_inline_keyhash** was set to TRUE (not the default value), the key-hash calculated on the *DataReader* for a dispose sample sent by a *DataWriter* was invalid. This led to a situation in which a disposed instance was not reported as such on the *DataReader* side.

This issue affected all language bindings except Java and the legacy .NET API. It also affected DynamicData.

This problem has been resolved.

[RTI Issue ID CORE-11378]

4.2.4 Deserialization of tampered/corrupted samples may have unexpectedly succeeded

A *DataReader* may not have detected that a truncated sample due to corruption or tampering was invalid. As a result, the application may have received samples with invalid content.

This issue has been resolved. Now, the deserialization of corrupted samples fails, and they are not provided to the application.

[RTI Issue ID CORE-11494]

4.2.5 Potential segmentation fault during batch sample serialization

Configuring batching with an unlimited value for **max_data_bytes** and a finite value for **max_samples** in the BATCH QoS Policy, in combination with setting the **dds.data_writer.history.memory_manager.fast_pool.pool_buffer_max_size** resource limit to 0, could have triggered a segmentation fault during batch sample serialization. Prior to the crash, the following message was logged:

```
!error serializing batch sample
```

This problem has been resolved. *Connext DDS* no longer issues a segmentation fault or logs an error message during batch sample serialization when using the configuration described above.

[RTI Issue ID CORE-11537]

4.2.6 Invalid serialization of samples with types containing nested structures with primitive members that require padding

In *Connext DDS* 6.0.1.20 and 6.1.0, the serialization of samples with a type containing one or more levels of nested complex types, where the nested types only had primitive members, may have failed. This means that a *DataReader* may have received an invalid value for a sample. For example:

```
struct MyType2 {
    long m21;
    long m22;
    double m23;
};

struct MyType {
    long m1;
    MyType2 m2;
};
```

This issue only applied when all of these conditions applied:

- You used XCDR1 data representation.
- The top-level type (MyType above) and the nested type containing only primitive members (MyType2 above) were appendable or final.
- There was a padding in the equivalent C/C++ type between the nested type member (m2 above) and the previous member (m1 above). In the above example, there is a 4-byte padding between m1 and m2 in MyType.

This problem affected DynamicData and the generated code for the following languages: C, C++, C++03, and C++11.

For generated code, a potential workaround to this problem was to generate code with a value of 1 or 0 for the **-optimization** parameter, but this may have had performance implications.

This problem has been resolved.

[RTI Issue ID CORE-11604]

4.2.7 When a sample in a batch could not be deserialized, it was not reported as lost

When a sample in a batch could not be deserialized, it was not reported as lost. This issue has been solved. Now, the DDS_SampleLostStatus is updated properly for deserialization issues.

[RTI Issue ID CORE-11923]

4.2.8 Deserialization failure when Java DataReader received compressed batch

A Java application running a *DataReader* that received a compressed batch ignored the batch compression, and the *DataReader* attempted to deserialize the compressed data, resulting in corrupted data or a deserialization error with the following log message:

```
"Exception caused by: not enough space available in the CDR buffer"
```

This problem has been fixed.

[RTI Issue ID CORE-12291]

4.2.9 Deserialization failure when DataReader received compressed sample with XCRD2 representation from Java DataWriter

A *DataReader* failed to deserialize samples if they were published by a Java *DataWriter* using compression and XCDR2 data representation.

This problem has been fixed.

[RTI Issue ID CORE-12357]

4.3 Fixes Related to Usability and Debuggability

4.3.1 Incompatible offered QoS in rtiddspy due to DDS_DataRepresentationQosPolicy

When using *rtiddspy*, you may have seen the following message:

```
Incompatible offered QoS on DataWriter associated with topic: "..."
```

This problem could be related to the QoS **DDS_DataRepresentationQosPolicy**, because *rtiddspy* was only using DDS_XCDR_DATA_REPRESENTATION. Now, however, *rtiddsspy* uses both XCDR and XCDR2 representations:

```
<representation>
    <value>
        <element>XCDR_DATA_REPRESENTATION</element>
        <element>XCDR2_DATA_REPRESENTATION</element>
    </value>
</representation>
```

The warning message will no longer appear, and *rtiddsspy* will no longer report an XCDR incompatibility.

[RTI Issue ID CORE-9372]

4.3.2 Failure enabling Heap Monitoring on VxWorks 6.6+

It was not possible to enable Heap Monitoring and create a *DomainParticipant* on VxWorks 6.6+. The following error messages were logged:

```
RTIOsapiThread_createTssFactory:TSS Factory can be created only once
DDS_DomainParticipantGlobals_initializeI:!create thread-specific storage factory
DDS_DomainParticipantFactory_newI:!create participant globals
DDSDomainParticipantFactory::create_instanceI:!create participant factory infrastructure
DDSDomainParticipantFactory::get_instance:!create participant factory
```

This issue has been resolved. Now you can enable Heap Monitoring and create *DomainParticipants* on VxWorks 6.6+.

[RTI Issue ID CORE-9674]

4.3.3 Instances resource leak in DataReader queue when instances were unregistered

If a *DataReader* received a message for an instance indicating that it was unregistered by a matching, remote *DataWriter*, and that instance was not already registered with the *DataReader*, the *DataReader* allocated an instance resource and never freed it. In order for an instance to be registered with a *DataReader*, it must have received a valid sample or dispose messages for that instance and must not have purged the instance based on the **autopurge_disposed_instances_delay** setting or due to automatic purging of empty instances in the NOT_ALIVE_NO_WRITERS instance state.

This issue caused the resource configured with **max_instances** to be consumed. If the value of **max_instances** was UNLIMITED, this issue caused an unbounded memory growth. If the value of **max_instances** was set to a finite value, then the limit may have been hit with no way to recover and reuse any of the used-up resources.

This issue has been resolved. Receiving an unregister message before any other sample for an instance no longer causes a resource to be consumed indefinitely.

[RTI Issue ID CORE-10762]

4.3.4 Large memory allocation on DataReaders due to tampered data samples

Corrupted or tampered data samples entailed large memory allocations up to a maximum of 2Gb of memory dynamically allocated. For example, this problem occurred when the length of an unbounded sequence or string was tampered with and set to a large number.

This problem has been fixed. Tampered samples will fail to be deserialized and they will not lead to large memory allocations.

[RTI Issue ID CORE-11344]

4.3.5 Waitset with the status SUBSCRIPTION_MATCHED did not work with Zero Copy transfer over shared memory

If you were using Zero Copy transfer over shared memory and a waitset with the status SUBSCRIPTION_MATCHED, the waitset was not triggered when there was a change in the status.

This issue has been fixed. Now the waitset works fine in combination with Zero Copy transfer over shared memory.

[RTI Issue ID CORE-11349]

4.3.6 "opening profiles group files" error

If you set several QoS files using the environment variable NDDS_QOS_PROFILES or the QoS setting **factory_qos.profile.url_profile**, the application sometimes failed with the following errors, even if the files existed:

```
export NDDS_QOS_PROFILES="file://file1.xml | file://file2.xml"
```

```
DDS_QosProvider_load_profiles_from_url_groupI:ERROR: opening profiles group files
'file://file1.xml | file://file2.xml'
[CREATE Participant] DDS_QosProvider_load_profiles_from_url_listI:ERROR: loading profiles
[CREATE Participant] DDS_QosProvider_load_profiles_from_env_varI:ERROR: loading profiles
[CREATE Participant] DDS_QosProvider_load_profilesI:ERROR: loading profiles
[CREATE Participant] DDS_DomainParticipantFactory_load_profilesI:!load profiles
[CREATE Participant] DDS_DomainParticipantFactory_create_participant_disabledI:ERROR: loading
profiles
```

These errors were caused by extra spaces or quotes in the list of file names.

Now, extra spaces or quotes will no longer cause these errors.

[RTI Issue ID CORE-11373]

4.3.7 Samples may have been lost if multiple readers were created in the same locator and push_on_write was set to false

If **push_on_write** was set to false in a *DataWriter's* DATA_WRITER_PROTOCOL QoS Policy, and multiple *DataReaders* were created in the same locator (same participant, same port), samples may have been lost.

This potential loss may have occurred when the second or subsequent *DataReader* was created. When the *DataWriter* detected the new *DataReader*, it sent an RTPS gap message, which may have gapped unsent samples. This was only an issue if **push_on_write** was set to false.

This problem has been resolved.

[RTI Issue ID CORE-11515]

4.3.8 DataWriter's send window did not behave correctly if min_send_window_size and max_send_window_size were not equal

A variable sized send window (configured by setting **DDS_RtpsReliableWriterProtocol_t::min_send_window_size** and **DDS_RtpsReliableWriterProtocol_t::max_send_window_size** to different values) had some issues.

On system startup, **DDS_DataWriterProtocolStatus::send_window_size** incorrectly returned the maximum size of the send window, when in fact the minimum was used. This was only an issue until the first **send_window_update_period** had elapsed, after which the correct value was reported.

If the send window was growing (i.e., the *DataWriter* had not received any NACK messages in the current **send_window_update_period**), the *DataWriter* would block based on the previous size of the send window.

The number of piggyback heartbeats sent with a sample was also incorrect, with the heartbeats never being sent in some cases.

These problems are resolved. The *DataWriter* now blocks based on the current send window size. The rate at which piggyback heartbeats are sent is calculated correctly. A piggyback heartbeat is now always sent when the send window size grows, avoiding the situation where the sending of the heartbeat was delayed due to a send window growing very fast.

[RTI Issue ID CORE-11529]

4.3.9 Potential DataWriter crash during remote DataReader reactivation

There was an issue that may have triggered a crash in a *DataWriter* using application-level acknowledgments. In particular, this issue may have triggered when a *DataWriter* transitioned a remote *DataReader* from inactive to active, which could have happened if the *DataReader* was temporarily unresponsive or too slow processing samples.

This problem has been resolved: a *DataWriter* no longer crashes when transitioning a remote *DataReader* from inactive to active.

[RTI Issue ID CORE-11802]

4.3.10 Possible decompression failure after receiving a compressed batch of an unkeyed data type

If a *DataReader* received a compressed batch where the type of the samples on the batch was unkeyed, the reader history kind was set to KEEP_LAST, and the reader resource limit **max_samples** was anything other than unlimited, a decompression failure was triggered with the following log message:

```
RTIOsapi_Zlib_uncompress:The input data was corrupted
RTICdrStream_uncompress:!uncompress sample
PRESPsReaderQueue_storeSampleToEntry:!uncompress stream
PRESPsReaderQueue_newData:!get entries
```

This issue has been solved. Now a valid compressed batch will not trigger this error.

[RTI Issue ID CORE-11830]

4.3.11 Potential unbounded memory growth during remote participant removal

A race condition issue may have prevented a *DomainParticipant* from completely removing remote participant resources.

When this issue triggered, the affected participant logged the following error:

```
PRESInterParticipant_removeRemoteParticipant:
[LDP=0x99843755,0x09333301,0x34442207,RE=0x09843755,0x99222201,0xED662207:0] Could not remove
remote endpoint
```

This problem is now resolved: the participant no longer logs errors nor leaks memory during remote participant removal.

[RTI Issue ID CORE-11842]

4.3.12 Unexpected exported symbols in Windows static libraries

In *Connext DDS* 6.1.0, static libraries wrongly exported symbols related to logging functions. In some scenarios, this could lead to linker errors (e.g., LNK2005) that were not present when trying to build against previous versions of *Connext DDS*.

This issue has been resolved.

[RTI Issue ID CORE-12008]

4.3.13 DomainParticipant memory usage increased significantly

The memory consumption of a *DomainParticipant* increased in release 6.1.0 by 512Kb or 1Mb if security was enabled. This issue has been resolved.

[RTI Issue ID CORE-12092]

4.3.14 Crash when enabling or disabling Network Capture

Your application may have crashed while enabling or disabling Network Capture. The crash may have occurred if other threads were concurrently initializing or finalizing either the TypeCodeFactory, the DomainParticipantFactory, or Network Capture. Instead of crashing, the previously mentioned actions may have failed or caused a memory leak. This issue has been resolved.

[RTI Issue ID CORE-12173]

4.3.15 Potential segmentation fault while logging a message on QNX systems (on PowerPC and Arm CPUs) and on INTEGRITY systems (on P4080 CPUs)

There was a potential segmentation fault while *Connext DDS* was logging a message. This problem was more likely to happen when another thread was concurrently unregistering a logging device. This problem affected *Connext DDS* 6.0.0 and above, as well as versions between 5.3.1.16 and 5.3.1.38 and versions 4.5d.rev41 and 4.5d.rev42. This problem affected QNX systems (only on PowerPC and Arm CPUs) and INTEGRITY systems (only on P4080 CPUs). This problem has been fixed.

[RTI Issue ID CORE-12214]

4.4 Fixes Related to Transports

4.4.1 Source code bundle missing critical piece of code related to UDP multicast support

In 6.1.0, the source code bundle was missing a critical piece of code related to UDP multicast. This issue has been resolved. The source code bundle is no longer missing code for UDP multicast support.

[RTI Issue ID CORE-11590]

4.4.2 Possible segmentation fault during TCP transport shutdown when using dynamic linking

If using dynamic linking, there may have been a segmentation fault when shutting down the TCP transport.

The stack trace of the thread crashing was the following:

```
#1  0x00007ffff6606c4f in RTIEventActiveGeneratorThread_loop (param=0xa159a0) at
/rti/jenkins/workspace/connextdds/6.1.0.0/x64Linux4gcc7.3.0/src/event.1.0/srcC/activeGenerator
/ActiveGenerator.c:397
#2  0x00007ffff6577dea in RTIOsapiThreadFactory_onSpawned (param=0xa19a70) at
/rti/jenkins/workspace/connextdds/6.1.0.0/x64Linux4gcc7.3.0/src/osapi.1.0/srcC/threadFactory/T
hreadFactory.c:211
#3  0x00007ffff65736d6 in RTIOsapiThreadChild_onSpawned (param=0xa19ab0) at
/rti/jenkins/workspace/connextdds/6.1.0.0/x64Linux4gcc7.3.0/src/osapi.1.0/srcC/thread/Thread.c
:1908
#4  0x00007ffff57c36db in start_thread (arg=0x7fffeeaa4700) at pthread_create.c:463
#5  0x00007ffff5afc71f in clone () at ../sysdeps/unix/sysv/linux/x86_64/clone.S:95
```

This problem has been resolved; the thread will no longer crash when closing the TCP transport.

[RTI Issue ID CORE-11625]

4.4.3 allow_interfaces_list and deny_interfaces_list could not have white spaces

Setting the properties **allow_interfaces_list** or **deny_interfaces_list** with white spaces did not work:

```
DDSPropertyQosPolicyHelper::add_property(dpQos.property,
"dds.transport.UDPv4.builtin.parent.allow_interfaces_list", "lo ", DDS_BOOLEAN_FALSE);
```

You would see an error similar to the following:

```
WARNING [0x0101FFE7,0x84C0B3AA,0x1F749265:0x000001C1{D=0}|CREATE DP|ENABLE|LC:DISC]DDS_
DomainParticipant_enableI:There are no valid locators for use by this participant. Please
validate that a valid transport is available for use by the participant and check your DDS_
TransportUnicastQosPolicy and DDS_TransportMulticastQosPolicy settings.
ERROR [0x0101FFE7,0x84C0B3AA,0x1F749265:0x000001C1{D=0}|CREATE DP|ENABLE|LC:DISC]DDS_
DomainParticipant_enableI:Automatic participant index failed to initialize. PLEASE VERIFY
CONSISTENT TRANSPORT / DISCOVERY CONFIGURATION.
[NOTE: If the participant is running on a machine where the network interfaces can change, you
should manually set wire protocol's participant id]
ERROR [0x0101FFE7,0x84C0B3AA,0x1F749265:0x000001C1{D=0}|CREATE DP] DDS_
DomainParticipantFactory_create_participant:ERROR: Failed to auto-enable entity
```

This issue has been resolved. Now, specifying a list of allowed or denied interfaces will work, even with white spaces in the property names.

[RTI Issue ID CORE-11770]

4.4.4 VxWorks kernel-mode shared memory didn't work after restarting an application

Starting with release 6.0.0, applications running in VxWorks kernel-mode could have problems discovering and communicating with other *DomainParticipants* if the original task that created the participant had exited, and the applications were re-started. These problems were due to some state that wouldn't be cleared when deleting the DomainParticipantFactory. This state is now being cleared.

[RTI Issue ID CORE-11933]

4.4.5 Applications using shared memory in VxWorks kernel-mode re-used a segment already in use

If an application in VxWorks used the shared memory transport, the *Connext DDS* libraries sometimes incorrectly assessed that a shared memory segment was stale and could be re-claimed, 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. This problem only occurred in VxWorks kernel-mode architectures. This problem has been fixed.

[RTI Issue ID CORE-11952]

4.4.6 Possible crash on Windows during UDP transport creation or update

In some cases, a *Connext DDS* application running on Windows could crash during the creation or update of a UDP transport if the logging verbosity was set to NDDS_CONFIG_LOG_VERBOSITY_STATUS_ALL for the NDDS_CONFIG_LOG_CATEGORY_COMMUNICATION category. This problem has been resolved.

[RTI Issue ID CORE-12037]

4.4.7 Valgrind issue when using allow or deny interfaces, UDPv6, and interface names

Valgrind™ reported uses of uninitialized values when setting any of the following properties to an interface name without wildcards:

- **dds.transport.UDPv6.builtin.parent.deny_interfaces**
- **dds.transport.UDPv6.builtin.parent.deny_multicast_interfaces_list**
- **dds.transport.UDPv6.builtin.parent.allow_interfaces**
- **dds.transport.UDPv6.builtin.parent.allow_multicast_interfaces_list**

This issue is now fixed. This Valgrind issue did not affect correctness and did not have an impact on functionality.

[RTI Issue ID CORE-12075]

4.5 Fixes Related to Content Filters and Query Conditions

4.5.1 Deletion of not-enabled ContentFilteredTopic failed with debug libraries

Trying to delete a ContentFilteredTopic that had been created in the disabled state incorrectly failed with a precondition error, unless the ContentFilteredTopic was enabled first. This problem only happened with the debug libraries, which checked an incorrect invariant condition.

This problem has been resolved.

[RTI Issue ID CORE-7232]

4.5.2 Unnecessary buffer allocation when using a QueryCondition

This issue was fixed in 6.1.0, but not documented at that time.

When using QueryConditions, an unnecessary buffer was allocated during the evaluation of a sample against the QueryCondition. This issue would have been noticeable in particular when the **max_serialized_size** of the data being evaluated was large or unbounded because the unnecessary buffer was being allocated to the **max_serialized_size** of the sample.

This issue only happened if the property **dds.data_reader.history.memory_manager.fast_pool.pool_buffer_max_size** was not set to a finite value and when using one of the DynamicData, Java, .Net or Traditional C++ APIs.

This behavior has been corrected and QueryCondition evaluation only allocates a buffer when necessary.

[RTI Issue ID CORE-8875]

4.5.3 DataReader may not have received expected samples from MultiChannel DataWriter

A *DataReader* using a ContentFilteredTopic may not have received expected samples from a MultiChannel *DataWriter*.

This issue occurred when the ContentFilteredTopic's expression on the *DataReader* and the channel expressions on the *DataWriter* used a MATCH operator on the same field and when any of the MATCH expression(s) contained negated intervals ([!]) . For example:

DataReader expression: myField MATCH AP1

DataWriter channel expression: myField MATCH *P[!2]

In this case, the *DataReader* should have received all samples published on the *DataWriter* channel, but this was not the case.

The issue has been resolved.

[RTI Issue ID CORE-11592]

4.5.4 Custom content filters unusable in certain cases

Custom content filters did not work in the following situations:

- In Modern C++, for any IDL type except when the code was generated with the now-removed **-legacyPlugin** option.
- In Traditional C++, for any IDL struct using inheritance.

Other language APIs were not affected.

This problem has been resolved.

[RTI Issue ID CORE-11622]

4.5.5 Significant performance degradation when using MultiChannel DataWriters

In release 6.1.0, you may have observed a significant performance degradation compared to previous releases when using MultiChannel *DataWriters*.

Release 6.1.0 introduced a regression in which a *DataReader* ended up subscribing to all the multicast addresses associated with a *DataWriter's* channels instead of subscribing to only the multicast addresses that can provide samples that pass the *DataReader* ContentFilteredTopic.

Note that this issue did not affect correctness, because the *DataReader* ended up filtering locally the samples that did not pass its ContentFilteredTopic expression.

This problem has been resolved.

[RTI Issue ID CORE-11742]

4.5.6 IP mobility event or interface disconnection may have led to increase in volume of repair traffic for DataReaders using ContentFilteredTopics

An IP mobility (change of IP address) or interface disconnection event on a subscribing application may have led to writer-side filtering being disabled for the *DataReaders* using ContentFilteredTopics in the subscribing application.

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*, the *DataWriter* sent a GAP protocol message to the *DataReader*. If the GAP message was lost, the *DataReader* NACKed the sample; instead of sending a new GAP message the *DataWriter* sent the sample.

This problem has been resolved.

[RTI Issue ID CORE-11774]

4.5.7 Subscribing application may have hung and consumed 100% CPU when publishing application used MultiChannel

A subscribing application may have hung and consumed 100% CPU when a matching publishing application used MultiChannel.

This problem only occurred when the filter expressions on the *DataReader* and one of the *DataWriters'* channels contained two overlapping intervals. For example:

DataReader: myField MATCH *[a-b]

DataWriter: myField MATCH P[a-c]

This problem has been fixed.

[RTI Issue ID CORE-11887]

4.5.8 ReadCondition may have incorrectly stayed enabled after a sample in the READ sample state was removed from the DataReader's queue

There were certain scenarios that caused a *ReadCondition* that was enabled when samples were in the READ_SAMPLE_STATE or ANY_SAMPLE_STATE to stay enabled forever. This would happen if a sample that had previously been read by the application was removed from the *DataReader's* queue, for example, due to KEEP_LAST history replacement.

This issue would cause any *Waitset* to which the *ReadCondition* was attached to immediately return successfully even though there were not actually any samples available in the *DataReader's* queue that matched the *ReadCondition*.

This issue only affected keyed data types. This issue has been fixed, and the *ReadConditions* will correctly be disabled once there are no more samples in the *DataReader's* queue that match its configured sample, view, and instance states.

[RTI Issue ID CORE-12168]

4.6 Fixes Related to Coherent Sets

4.6.1 Group coherent sets incorrectly reported as incomplete

In 6.1.0, a group coherent set for which all the samples were received may have been erroneously reported as incomplete (**SampleInfo.coherent_set_info.incomplete_coherent_set** was set to true). This issue only occurred when batching was used in any of the group's *DataWriters*. This problem has been fixed.

[RTI Issue ID CORE-11870]

4.6.2 Rejected reason and count may have been incorrect when sample was rejected using batching and coherent set

The rejected reason and count may have been incorrect when a sample was part of a batch or coherent set. The problem occurred only if the number of samples in the batch was greater than the maximum number of samples per remote *DataWriter*.

This issue have been resolved. Now the rejected reason and count are updated properly for samples in batches or coherent sets.

[RTI Issue ID CORE-12033]

4.6.3 Segmentation fault when coherent set finalized while using destination order by source timestamp

A *DataWriter* in a publishing application may have experienced a segmentation fault when all these conditions were true:

- The *DataWriter* was configured with **destination_order.kind** set to BY_SOURCE_TIMESTAMP_DESTINATIONORDER_QOS.
- The *DataWriter* was configured with **destination_order.scope** set to INSTANCE_SCOPE_DESTINATIONORDER_QOS.
- The *DataWriter* tried to publish a coherent set.

The segmentation fault occurred when the coherent set was finalized.

This problem has been fixed.

[RTI Issue ID CORE-12197]

4.7 Fixes Related to Dynamic Data

4.7.1 Binding to an unset member of a union in a DynamicData object and unbinding without setting a value led to a crash or error messages

When using the DynamicData API, in order to set and access nested members of a DynamicData object, you must first bind[1] to the nested members. If the type being used by a DynamicData object contained a union, the binding to an unset member of the union, and then unbinding without setting a value, led to a crash or error messages similar to the following:

[1]In some language APIs, such as Modern C++, binding is referred to as loaning.

```
RTI0x2000027:!precondition: "memManager == ((void *)0)"
```

This issue has been fixed. Now it is safe to bind and unbind to any member in a union without setting a value for that member.

[RTI Issue ID CORE-11662]

4.8 Fixes Related to DDS API

4.8.1 Could not create a QosProvider with custom QosProviderParams

Previous releases didn't provide a way to create a QosProvider with custom QosProviderParams. This option, which allows disabling the loading of QoS profiles from certain default locations, was only provided to configure the Default QosProvider (QosProvider::Default). A new standalone extension function, **rti::core::create_qos_provider_ex**, allows creating a QosProvider with custom QosProviderParams.

[RTI Issue ID CORE-7232]

4.8.2 Missing SampleInfo.ReceptionTimestamp property (C# API only)

The new Connext C# API introduced in 6.1.0 did not provide the **SampleInfo.ReceptionTimestamp** extension property. This property has been added and is now available.

[RTI Issue ID CORE-11908]

4.8.3 Possible memory corruption when batching and compression enabled in Java application

A *DataWriter* running in a Java application with compression and batching enabled could cause memory corruption with undefined behavior in the publishing application.

This problem has been fixed.

[RTI Issue ID CORE-12436]

4.8.4 DDS_Int8 incorrectly mapped to unsigned value on QNX systems (on PowerPC and Arm CPUs) and on INTEGRITY systems (on P4080 CPUs)

DDS_Int8 was incorrectly mapped to an unsigned value on QNX systems (only on PowerPC™ and Arm CPUs) and on INTEGRITY systems (only on P4080 CPUs). This problem has been fixed. DDS_Int8 is now mapped to a signed value on all platforms.

[RTI Issue ID CODEGENII-1639]

4.9 Fixes Related to Modern C++ API

In addition to 4.8 Fixes Related to DDS API on the previous page, this release includes the following fixes, which are specific to the Modern C++ API.

4.9.1 find_datawriters, find_readers, and find_topics did not work for XML-defined DynamicData entities

When a *DataReader*, *DataWriter*, or *Topic* was created from its XML definition (via **QosProvider::create_participant_from_config**), any lookup function that returned AnyDataReader, AnyDataWriter, or AnyTopic failed. It was required to use a lookup function returning the typed entity (DataReader<T>, DataWriter<T>, Topic<T>).

This problem has been partially fixed in this release. When the type T is DynamicData, it is now possible to use the functions returning the "Any" entities.

Note that this problem only affected entities created from XML, not entities created using their respective constructors.

[RTI Issue ID CORE-10940]

4.9.2 Missing DataReader constructor

The previous release added constructors that receive listeners as **shared_ptr**'s. However, a constructor receiving both a ContentFilteredTopic and a **shared_ptr** to the listener was not added.

This missing constructor has been added now.

[RTI Issue ID CORE-11594]

4.9.3 Missing symbols for DataReaderResourceLimitsInstanceReplacementSettings on Windows

In 6.1.0, the symbols for **DataReaderResourceLimitsInstanceReplacementSettings** were missing in the Modern C++ libraries on Windows systems. This caused errors building applications that linked against them. This issue has been resolved and now the symbols are properly exported.

[RTI Issue ID CORE-11646]

4.9.4 Missing forward declarations in some header files

The Modern C++ API contains a number of "<namespace>fwd.hpp" headers that provide forward declarations for all its types. Some types (Publisher, LoanedSamples, QosProvider) were not forward-declared. This problem has been resolved.

[RTI Issue ID CORE-11651]

4.9.5 Compilation errors when VxWorks application used Boost and Modern C++ API in same source file

The Modern C++ API internally uses a subset of Boost 1.61. All the Boost symbols have been renamed to avoid collisions with user applications that also include Boost. However some standard functions that are missing from VxWorks are defined in Boost headers as inline functions (symlink, readlink, times, truncate). Source files that include Boost and the *Connext DDS* Modern C++ API may have failed to compile due to duplicate symbols, because these functions are defined both in the Boost headers used by the Modern C++ API and the user-included headers.

To avoid these errors, you will need to perform one of the following options:

- Make sure the Boost headers are included *before* any RTI header. For example:

  ```
  #include <boost/shared_ptr.hpp> // FIRST
  ...
  #include <dds/domain/DomainParticipant.hpp> // SECOND
  ...
  ```

  The RTI Boost headers will detect that another Boost installation has been included, and will exclude the conflicting symbols.

- Compile the source files that use Boost with the option **-DRTI_USE_BOOST**. The RTI Boost headers will recognize this preprocessor definition and exclude the conflicting symbols.

[RTI Issue ID CORE-11656]

4.9.6 Possible race condition between WaitSet::dispatch and detach_condition

When a thread called **dispatch()** at the same time that another thread called **detach_condition()** on the same *WaitSet* object, in the Modern C++ API only, there was a possibility of a race condition leading to undefined behavior (likely a crash due to null dereference).

This problem has been resolved. It's now safe to call **detach_condition()** while that condition is being dispatched.

[RTI Issue ID CORE-11800]

4.9.7 Possible segmentation fault when receiving samples containing wstrings

In releases 6.0.1.22 and higher and in release 6.1.0.3, a subscribing application may have crashed when receiving samples containing wstrings. This issue occurred when the following three conditions were all true:

- The language binding was modern C++.
- The size of wchar_t was 4-byte.
- The length of a wstring member in the sample was equal to the maximum allowed. For example:

```
struct MyType {
     wstring<5> m1;
};
```

For this type, the deserialization of a sample with the following value for m1 would lead to a segmentation fault: L"Hello".

This issue has been fixed.

[RTI Issue ID CORE-11896]

4.10 Fixes Related to XML Configuration

4.10.1 XSD schema enforced strict ordering for elements in <publisher> and <subscriber> tags

The XSD schema **rti_dds_profiles_definitions.xsd** enforced a strict ordering of elements in the <publisher> and <subscriber> tags. Because of this, the XSD validation showed an error when the <data_writer> or <data_reader> element was set after the <publisher_qos> or <subscriber_qos> element, respectively. This problem has now been resolved. The ordering of elements under <publisher> and <subscriber> tags no longer matters for XSD validation.

[RTI Issue ID CORE-9374]

4.10.2 rtiddsping RTIDDSPING_QOS_PROFILES.example.xml file had wrong Durability QOS

The *rtiddsping* RTIDDSPING_QOS_PROFILES.example.xml file had the wrong *DataWriter-/DataReader* Durability QoS. It contained:

```
<durability>
    <kind>TRANSIENT_LOCAL_DURABILITY_QOS</kind>
</durability>
```

It should have been:

```
<durability>
    <kind>VOLATILE_DURABILITY_QOS</kind>
</durability>
```

This issue has been fixed, and now it shows the correct durability.

[RTI Issue ID CORE-11253]

4.10.3 XML parsing failed for DataReaderResourceLimitsInstanceReplacementSettings

The XSD for the **DataReaderResourceLimitsInstanceReplacementSettings** correctly indicated that the valid values for the **alive_instance_replacement**, **disposed_instance_replacement**, and **no_writers_instance_replacement** fields are as follows:

- NO_INSTANCE_REMOVAL
- EMPTY_INSTANCE_REMOVAL
- FULLY_PROCESSED_INSTANCE_REMOVAL
- ANY_INSTANCE_REMOVAL

However, if any of these values was used, the XML parser would fail to parse it because the parser was expecting it to be prefixed with 'DDS_'. An error similar to the following was shown:

```
DDS_DataReaderInstanceRemovalKind_parse:!parse 'EMPTY_INSTANCE_REMOVAL'
DDS_XMLQos_onEndDataReaderResourceLimitsElement:Parse error at line 83: The value associated to
the tag 'alive_instance_removal' is not valid
```

This issue has been fixed. Now the values listed above are correctly parsed. The same values prefixed with 'DDS_' are also still accepted by the XML parser, but are not considered valid according to the XSD schema and therefore will no longer show up as valid options when using autocomplete in an IDE.

[RTI Issue ID CORE-11535]

4.10.4 Property validation failed when setting custom alias for builtin transport

According to "Installing Additional Builtin Transport Plugins with PropertyQosPolicy" in the *RTI Connext DDS Core Libraries User's Manual*, it is possible to set up, through XML, custom aliases for the builtin transports:

```
<domain_participant_qos>
    <transport_builtin>
        <mask>MASK_NONE</mask>
    </transport_builtin>
    <property>
        <value>
            <element>
                <name>dds.transport.load_plugins</name>
                <value>dds.transport.UDPv4.mytransport</value>
            </element>
            <element>
                <name>dds.transport.UDPv4.mytransport.aliases</name>
                <value>CustomUDP</value>
            </element>
        </value>
```

```
    </property>
</domain_participant_qos>
```

However, the property validation didn't recognize the "aliases" property. When using the property aliases for a builtin transport, the property validation failed:

```
[0x0101816F,0x6C2511A0,0x0CBF3229:0x000001C1{N=helloworldParticipant,D=0}|CREATE DP|ENABLE]
DDS_PropertyQosPolicy_validate_plugin_property_suffixes:Unexpected property:
dds.transport.UDPv4.mytransport.aliases. Closest valid property:
dds.transport.UDPv4.mytransport.multicast_ttl. If you wish to proceed with this property name
anyway, change 'dds.transport.UDPv4.mytransport.property_validation_action' to 'VALIDATION_
ACTION_SKIP' or 'VALIDATION_ACTION_WARNING'.
```

This problem has been resolved. Now you can use the "aliases" property.

[RTI Issue ID CORE-11581]

4.10.5 XML parser crashed from infinite recursion when XML QoS configuration contained inheritance loop

An inheritance loop was formed when a <qos_profile> inherited from itself or when any <xxx_qos> inherited from itself or its encapsulating <qos_profile>. Inheritance can be performed by using the base_name attribute or <base_name> tag.

In the previous release, the XML parser would crash when the XML QoS configuration contained an inheritance loop. This problem has been resolved. If the parser detects an inheritance loop, it now throws an error.

[RTI Issue ID CORE-11731]

4.11 Fixes Related to Vulnerabilities

4.11.1 Fixes related to Connext DDS

This release fixes some potential vulnerabilities, including RTI Issue IDs CORE-11599, CORE-11649, CORE-11712, CORE-11749, CORE-11750, CORE-11751, CORE-11773, CORE-11882, CORE-11885, CORE-12380, COREPLG-568, and COREPLG-571.

4.11.2 Fixes related to third-party dependencies

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

4.11.2.1 Potential arbitrary code execution in Connext DDS application upon parsing of ContentFilteredTopics, QueryConditions, or TopicQuery filters due to vulnerabilities in Flex

The Core Libraries filter parser had a third-party dependency on Flex version 2.5.31. That version of Flex is known to be affected by a number of publicly disclosed vulnerabilities.

These vulnerabilities have been fixed by upgrading to the latest stable version of Flex, 2.6.4. See "Third-Party Software Upgrades" in RTI Connext DDS Core Libraries What's New in 6.1.1.

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

- With Connext Secure (enabling RTPS protection):
  - Exploitable through a compromised local file system containing an XML configuration file with a malicious filter.
  - Application could crash or leak sensitive information. An attacker could execute code with *Connext DDS* application privileges.
  - 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 an XML configuration file with a malicious filter.
  - Remotely exploitable through malicious RTPS messages.
  - Application could crash or leak sensitive information. An attacker could execute code with *Connext DDS* application privileges.
  - 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-12336]

4.11.2.2 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 2.2.5. That version of Expat is known to be affected by a number of publicly disclosed vulnerabilities.

These vulnerabilities have been fixed by upgrading to the latest stable version of Expat, 2.4.4. See "Third-Party Software Upgrades" in RTI Connext DDS Core Libraries What's New in 6.1.1.

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

- With Connext Secure (enabling RTPS protection):
  - Exploitable through a compromised local file system containing malicious XML/DTD files.

  - Application could crash or leak sensitive information.
  - CVSS v3.1 Score: 6.8 MEDIUM
  - CVSS v3.1 Vector: AV:L/AC:L/PR:N/UI:N/S:U/C:L/I:N/A:H

- Without Connext Secure:
  - Exploitable through a compromised local file system containing malicious XML/DTD files.
  - Remotely exploitable through malicious RTPS messages.
  - Application could crash or leak sensitive information.
  - CVSS v3.1 Score: 8.2 HIGH
  - CVSS v3.1 Vector: AV:N/AC:L/PR:N/UI:N/S:U/C:L/I:N/A:H

[RTI Issue ID CORE-12340]

4.11.3 Vulnerability assessments

Some *Connext DDS* components, including the Modern C++ API, use the following boost 1.61 modules only: align/ config/ core/ exception/ functional/ mpl/ predef/ preprocessor/ smart_ptr/ type_traits/ typeof/ utility/.

These modules are not currently affected by publicly disclosed vulnerabilities.

4.12 Other Fixes

4.12.1 Negative duration passed to Waitset.wait

If you passed Waitset.wait a negative duration, undefined behavior occurred. Now, when you pass a negative duration, BAD_PARAMETER is thrown.

[RTI Issue ID CORE-8236]

4.12.2 Malformed samples with invalid strings not dropped by DataReader in C, traditional C++, and modern C++

A *DataReader* may have provided the application a malformed sample containing an invalid value (not null-terminated) for a string member. The string member may not have been null-terminated, resulting in undefined behavior if the application tried to access it.

This issue has been addressed. The *DataReader* will fail to deserialize the sample, and the sample will not be provided to the application.

[RTI Issue ID CORE-11203]

4.12.3 Float and double ranges may not have been enforced correctly

Float and double ranges may not have been enforced correctly. Float and double member values that should not have passed the check ended up passing it.

This issue only occurred under any of the following conditions:

- For "float":
  - In all languages but Java, when @min was set to -3.4E38 for a member, a value smaller than @min passed the check when it should not have.
  - In all languages but Java, when @max was set to 3.4E38 for a member, a value greater than @max passed the check when it should not have.
- For "double":
  - In all languages but Java, when @min was set to -1.7E+308 for a member, a value smaller than @min passed the check when it should not have.
  - In all languages but Java, when @max was set to 1.7E+308 for a member, a value greater than @max passed the check when it should not have.
- For "float" and "double":
  - In all languages but Java, when the member value was set to INFINITY, samples passed the range check when they should not have.
  - In all languages, when the member value was set to NaN, samples passed the range check when they should not have.

This problem has been resolved.

[RTI Issue ID CORE-11358]

4.12.4 Linking static Linux or QNX libraries with object files built with -fPIC failed

In 6.1.0, RTI no longer built Linux or QNX static libraries with the **-fPIC** flag (whereas previously RTI did build with this flag). Therefore, you could not link Linux or QNX static libraries in 6.1.0 with object files built with the **-fPIC** flag. You likely ran into this problem if you used the static Linux or QNX library in 6.1.0 while building a shared library.

This problem has been fixed. RTI now builds these libraries with the **-fPIC** flag again.

[RTI Issue ID PLATFORMS-2519]

Chapter 5 What's Fixed in 6.1.0

Release 6.1.0 is a general access release based on the maintenance release 6.0.1. This section describes bugs fixed in 6.1.0. These fixes have been made since 6.0.1.

5.1 Fixes Related to Discovery

5.1.1 DataReader DDS_LIVELINESS_CHANGED_STATUS may not have worked properly

Connext DDS may have reported an incorrect DDS_LIVELINESS_CHANGED_STATUS for a *DataReader* in the following scenarios:

Multiple DataReaders

Consider a *DomainParticipant* with several *DataReaders* all matching a *DataWriter*. If *DataWriter* liveliness was lost for one of the *DataReaders*, the *DataReader* callback **on_liveliness_changed** was not called.

MultiChannel or TopicQuery

Consider a *DataWriter* and a *DataReader* using MultiChannel or TopicQueries. If the *DataWriter* liveliness changed (it was either lost or recovered) for the *DataReader*, the callback **on_liveliness_changed** was called, but it may have provided an incorrect **last_publication_handle**.

Both of these scenarios have been fixed. Now the callback **on_liveliness_changed** is called when expected, and it matches the correct *DataWriter's***last_publication_handle**.

[RTI Issue ID CORE-7626]

5.1.2 Potentially wrong deserialization of vendor-specific BuiltinTopicData fields

In the Simple Endpoint Discovery process:

- During serialization: the vendorId was set after some vendor-specific BuiltinTopicData fields were set.
- During deserialization: the vendorId was initialized to RTI_Vendor (0x0101) until it was deserialized.

This behavior could lead to issues:

- If the remote non-RTI implementation sent other vendor-specific fields before it sent the vendorId, *Connext DDS* processed those fields as RTI fields. But those fields, sent from another vendor, might have a different meaning.
- Likewise, if the other vendor used a logic similar to RTI's (serializing and deserializing vendor-specific fields based on the vendorId), then this vendor would also process the vendor-specific fields incorrectly when receiving them from *Connext DDS*.

This issue has been resolved.

- For serialization: *Connext DDS* now serializes the vendorId before any other vendor-specific fields.
- For deserialization: *Connext DDS* now derives the vendorId from the RTPS header if it has not parsed the vendorId yet.

[RTI Issue ID CORE-9755]

5.1.3 Discovery issues when reusing shared memory segments

Connext DDS tries to reuse shared memory segments that were already allocated if the process that owned them is not running anymore. This is normal behavior.

Reusing a shared memory segment, however, sometimes led to discovery issues if the shared memory **host_id** of the application was different than the one stored in the segment. (The shared memory **host_id** is computed based on the values of the **wire_protocol rtps_auto_id_kind** and **rtps_host_id**.)

This problem has been fixed. This issue was a regression introduced in *Connext DDS* 6.0.0. It affected only 6.0.x releases.

[RTI Issue ID CORE-10065]

5.1.4 DomainParticipant announcement lost after IP mobility event

In some operating systems *Connext DDS* can detect a new network interface before a socket can send a packet through it. This situation leads to the loss of the *DomainParticipant* announcement related to this IP mobility event and delays the notification to other *DomainParticipants* in this new network until the next periodic announcement.

To prevent this problem, a new property has been added: **dds.domain_participant.network_interface_event_notification_delay**. This property takes an integer value between 0 and 60000 and delays the *DomainParticipant* announcements that include a new interface for that amount of milliseconds.

The default value of **dds.domain_participant.network_interface_event_notification_delay** is 0 (no delay is applied). The value required to solve the issue will depend on the operating system and network devices involved. A value too small may not prevent the issue. A value too large may not improve the performance.

[RTI Issue ID CORE-10402]

5.1.5 Unexpected errors when IP mobility event triggered during DomainParticipant enabling

There was a rare race condition that may have triggered unexpected errors during *DomainParticipant* enabling or deletion. In particular, this issue may have been triggered if there was a change in the local host network interfaces at the same time the *DomainParticipant* was being enabled.

When this issue was triggered, the *DomainParticipant* may have shown errors during its enabling. The errors shown were similar to the following:

```
[0X10146B6,0X25F8B60B,0X33EC3594:0|UPDATING WAN INTERFACE ADDRESSES]
Participant.c:1895:PRESParticipant_compareImmutableProperty:!equal property: builtin endpoint
mask
[0X10146B6,0X25F8B60B,0X33EC3594:0|UPDATING WAN INTERFACE ADDRESSES]
DomainParticipantPresentation.c:2249:DDS_DomainParticipantPresentation_update_participant_
locatorsI:ERROR: Failed to set participant QoS
[0X10146B6,0X25F8B60B,0X33EC3594:0|UPDATING WAN INTERFACE ADDRESSES]
DomainParticipant.c:16246:DDS_DomainParticipant_update_participant_locatorsI:Failed to update
locators: participant locators
[0X10146B6,0X25F8B60B,0X33EC3594:0|UPDATING WAN INTERFACE ADDRESSES]
DomainParticipant.c:16707:DDS_DomainParticipant_onNetworkInterfaceChanged:Failed to update
locators: update participant locators
```

Another consequence of this issue was the *DomainParticipant* showing errors during its deletion. The errors shown were similar to the following:

```
[DELETE Participant] Receiver.c:1839:RTINetioReceiver_preShutdownWakeup:unremoved EP
```

This issue has been fixed. The *DomainParticipant* will no longer show unexpected errors if there is a network interface change at the same time the *DomainParticipant* is being enabled.

[RTI Issue ID CORE-10637]

5.1.6 Incorrect start time for event that checks for remote participant liveliness

The start time for the event that checks for remote participant liveliness was not the one configured by **DDS_DiscoveryConfigQosPolicy::max_liveliness_loss_detection_period**. (The **max_liveliness_loss_detection_period** is the maximum amount of time between when a remote entity stops maintaining its

liveliness and when the matched local entity realizes that fact. You can find out that a remote participant has lost liveliness by listening to the Participant Built-in discovery data.)

The start time for this event is now correct.

[RTI Issue ID CORE-10732]

5.2 Fixes Related to Usability and Debuggability

5.2.1 DDS_DataWriter::get_matched_subscription_data returned data that had not been applied yet

Information about *DataReaders* is communicated using the SubscriptionBuiltinTopicData builtin discovery channel. Changing certain properties of a *DataReader* causes new subscription data to be propagated to matching *DataWriters* via this channel (e.g., content filter, partition, or deadline changes). Applications can retrieve this subscription data with the **DataWriter::get_matched_subscription_data** API. In previous releases, this API may have returned data that had not yet taken effect in the *DataWriter*. This meant that it was not possible to make any decisions in the application based on the returned data.

For example, some applications may have waited for a content filter expression to be updated before beginning to publish data that matched the *DataReader's* most up-to-data filter expression. Before this issue was fixed, it was possible to see the *DataReader's* updated filter expression before the *DataWriter* started to use it for writer-side filtering. Therefore, any samples that were written based on the filter expression in the returned subscription data before the filter was applied to the *DataWriter* may have been filtered out by the *DataWriter*.

This issue has been resolved when the API is called outside of a listener callback. This API has been updated to block until the most recent changes known to the *DataWriter* have been applied. The **DataWriter::get_matched_subscription_data** will no longer return data that has not yet taken effect in the *DataWriter*. When called inside of a listener callback, it is still possible for the aforementioned issue to occur. The recommended pattern for usage of this API then is to wait for subscription data to be received either through polling this API or by installing a listener on the SubscriptionBuiltinTopicData builtin *DataReader*. When a new sample is received by the builtin *DataReader*, the **DataWriter::get_matched_subscription_data** may be called in a separate thread and will return the expected matched subscription data once it has been applied to the *DataWriter*.

Because the **DataWriter::get_matched_subscription_data** API blocks, it is possible for this API to time out while waiting for the changes to be applied. A timeout may happen if the *DataReader's* subscription data is changing rapidly, preventing the *DataWriter* from returning valid information before newer data has been received, or if an application is performing a task in a listener callback, thereby preventing the middleware's threads from executing events in a timely manner.

[RTI Issue ID CORE-5821]

5.2.2 last_reason field in DDS_SampleLostStatus contained invalid value if sample lost using Best Effort

If a sample was lost using DDS_BEST_EFFORT_RELIABILITY_QOS, the field **last_reason** in DDS_SampleLostStatus may have contained an invalid value, and you would have seen the following error message:

```
DDS_SampleLostStatus_from_presentation_status:ERROR:Fail to get SampleLostStatus (unknown kind)
```

This problem has been resolved. Now every time a sample is lost using DDS_BEST_EFFORT_RELIABILITY_QOS, the field **last_reason** will be correct.

[RTI Issue ID CORE-7281]

5.2.3 Eventual consistency not guaranteed when using DestinationOrderQosPolicy kind BY_SOURCE_TIMESTAMP and original writer sample identities

Systems that require an eventual consistency guarantee must use the DestinationOrderQosPolicyKind BY_SOURCE_TIMESTAMP. However, in cases where original writer sample identities were being used, eventual consistency was not guaranteed.

Original writer sample identities are used by *Routing Service* and *Persistence Service* to write samples on behalf of other *DataWriters*. Samples coming from these services include information about the service's physical *DataWriter* as well as the original *DataWriter* for the sample. In a situation in which one *DataReader* was receiving samples from the original *DataWriter,* and another *DataReader* was receiving samples from either the *Routing Service* or *Persistence Service*, it was possible that the two *DataReaders* would end up with different final values. This could happen because when two samples have the same source timestamp, the *DataWriter's* GUID is used to determine which of the samples to keep in the *DataReader*. The *DataReader* that was receiving samples directly from the original *DataWriter* would keep both samples while the *DataReader* that was receiving the samples through a service would drop the second sample if the service *DataWriter's* GUID had a lower value than the original *DataWriter's* GUID.

To fix this, the original *DataWriter's* GUID is now used to break ties when two consecutive source timestamps are equal, as opposed to the physical *DataWriter's* GUID.

[RTI Issue ID CORE-9792]

5.2.4 Piggyback heartbeats may not have been sent with batching

A *DataWriter* using batching may not have sent piggyback heartbeats, or it may have sent them at the wrong rate if **max_send_window_size** was set to UNLIMITED and **max_batches** was set to a finite value.

This problem has been resolved.

[RTI Issue ID CORE-9801]

5.2.5 Two crashing threads prevent the backtrace from being printed

When several threads crashed at the same time, the backtrace was not logged because the second crash exited the application before the first thread could print the backtrace.

This problem is resolved. Now when several threads crash at the same time, the backtrace of each of them is logged.

[RTI Issue ID CORE-9895]

5.2.6 DNS Tracker thread name not logged properly

The thread name logged for the DNS Tracker when enabling the **NDDS_Config_LogPrintFormat NDDS_CONFIG_LOG_PRINT_FORMAT_VERBOSE** print format was not correct.

Now, the correct name is always logged for the DNS Tracker logging messages.

[RTI Issue ID CORE-9899]

5.2.7 Backtrace not available when using library that was linked dynamically

If a crash occurred in a library that was linked dynamically, the backtrace did not provide useful information. For example:

```
Backtrace:
        #1      ?? ??:0 [0x7EB8347D]
        #2      ?? ??:0 [0x7F2E75D0]
        #3      ?? ??:0 [0x7CA5BA56]
        #4      ?? ??:0 [0x7CA793C8]
        #5      ?? ??:0 [0x7CA7B8A3]
                #6      ?? ??:0 [0x7E056B45]
        #7      ?? ??:0 [0x7E226E74]
        #8      ?? ??:0 [0x7E22983E]
        #9      ?? ??:0 [0x7EB84FB3]
        #10     ?? ??:0 [0x7F2DF5F0]
        #11     ?? ??:0 [0x7FF4C84D]
Segmentation fault (core dumped)
```

This problem has been fixed. Now the backtrace provides all the available information when the crash is in a library linked dynamically.

[RTI Issue ID CORE-9939]

5.2.8 Heap monitoring logging error "inconsistent free/alloc" could lead to unexpected behavior

The following log message was not formed correctly.

```
RTIOsapiHeap_freeMemoryInternal: inconsistent free/alloc: block id %#X being freed with %s and
was allocated with %s
```

It was missing one of the variadic arguments needed for the format specifiers of the format. The missing argument could lead to a segmentation fault on certain architectures.

For example, on the architecture x64Darwin15clang7.0, heap monitoring reported the following error:

```
thread #1: tid = 0x2882084, 0x00007fff86852d32 libsystem_c.dylib`strlen + 18, queue =
'com.apple.main-thread', stop reason = EXC_BAD_ACCESS (code=1, address=0x185fbfbf30)
* frame #0: 0x00007fff86852d32 libsystem_c.dylib`strlen + 18
frame #1: 0x00007fff868986e8 libsystem_c.dylib`__vfprintf + 5713
frame #2: 0x00007fff868c135d libsystem_c.dylib`__v2printf + 669
frame #3: 0x00007fff868a55a9 libsystem_c.dylib`_vsnprintf + 596
frame #4: 0x00007fff868a565e libsystem_c.dylib`vsnprintf + 80
frame #5: 0x000000010198d19f libnddscored.dylib`RTILog_vsnprintf(str="inconsistent free/alloc:
block id 0X106651E being freed with \"RTIOsapiHeap_unknownFunction _ , size=999,
format="inconsistent free/alloc: block id %#X being freed w
ith \"%s\" and was allocated with \"%s\"\n", args=0x00007fff5fbfbed0) + 111 at Log.c:109
```

This problem has been resolved.

[RTI Issue ID CORE-10120]

5.2.9 Incorrect number of lost samples reported when using Best Effort and batching

When a batch of samples was lost using DDS_BEST_EFFORT_RELIABILITY_QOS, the counters of DDS_SampleLostStatus were not updated correctly. They incremented the number of batches lost and not the number of samples.

This has been resolved. Now when a batch is lost, the number of samples in the batch is updated properly in DDS_SampleLostStatus.

[RTI Issue ID CORE-10151]

5.2.10 Unexpected log message when calling DataWriter::get_matched_subscription_data or DataReader::get_matched_publication_data on unmatched endpoints

The **DataWriter::get_matched_subscription_data** and **DataReader::get_matched_publication_data** APIs return RETCODE_PRECONDITION_NOT_MET when called using subscription or publication handles of endpoints that do not match with the calling endpoint. This is normal operation for the API and should not produce any logging messages at the exception log level; however, starting in release 6.0.0, an exception was printed in this case. This issue has been fixed. The log message is now printed at the warning log level, as was the case in releases previous to 6.0.0.

[RTI Issue ID CORE-10163]

5.2.11 Number of bytes reported in protocol statistics did not represent RTPS protocol bytes sent on wire

Previously, some of the protocol statistics that were measured in bytes (such as **sent_heartbeat_bytes**) represented the number of RTPS protocol message bytes sent on the wire, while other protocol statistics (such as **received_sample_bytes**) represented the size of the payload. Now all protocol statistics report the number of bytes in the RTPS protocol messages sent on the wire.

[RTI Issue ID CORE-10215]

5.2.12 last_instance_handle in DDS_SampleRejectedStatus, for keyed data in batches, may not have been correct

Previously when using batching and keyed data, if samples were rejected, the last_instance_handle field in DDS_SampleRejectedStatus may not have been correct. This problem has been resolved. Now every time there are rejected samples, the value of last_instance_handle in DDS_SampleRejectedStatus is correct.

[RTI Issue ID CORE-10405]

5.2.13 Unexpected property: com.rti.serv.secure.internal_plugin_context (error message)

The following error message may have been triggered when running *RTI Security Plugins* using an evaluation license:

```
DDS_PropertyQosPolicy_validate_plugin_property_suffixes:Unexpected property:
com.rti.serv.secure.internal_plugin_context. Closest valid property:
com.rti.serv.secure.openssl_engine
RTI_Security_PluginSuite_create:Inconsistent QoS property: com.rti.serv.secure.
DDS_DomainParticipantTrustPlugins_initialize:!create security plugin
DDS_DomainParticipant_createI:!create builtin trust plugins support
DDS_DomainParticipantFactory_create_participant_disabledI:!create participant
```

This issue has been resolved. Now the Security Plugin will be created and the error message will not appear.

[RTI Issue ID CORE-10472]

5.2.14 DDS_DataWriterProtocolStatus.pushed_sample_count for a DataWriter may have been incorrect when data was sent to multiple locators

If a *DataWriter* was sending data to multiple locators and data fragmentation was not used, the **pushed_sample_count** and **pushed_sample_bytes** statistics may have been incorrect. These fields within the DDS_DataWriterProtocolStatus may have shown that only a single sample had been sent, even though multiple RTPS packets had been put on the wire. This problem has been resolved.

[RTI Issue ID CORE-10490]

5.2.15 Unexpected property: com.rti.serv.secure.openssl_engine.[engineName].[cmdName]

The following error message may have been triggered when running *RTI Security Plugins*.

```
DDS_PropertyQosPolicy_validate_plugin_property_suffixes:Unexpected property:
com.rti.serv.secure.openssl_engine.[engineName].[cmdName]. Closest valid property:
com.rti.serv.secure.openssl_engine
RTI_Security_PluginSuite_create:Inconsistent QoS property: com.rti.serv.secure.
DDS_DomainParticipantTrustPlugins_initialize:!create security plugin
DDS_DomainParticipant_createI:!create builtin trust plugins support
DDS_DomainParticipantFactory_create_participant_disabledI:!create participant
```

This issue has been resolved. Now the Security Plugin will be created and the error message will not appear.

[RTI Issue ID CORE-10535]

5.2.16 Potential deadlock in rare error conditions

There were a few error conditions when enabling a *DomainParticipant* or asserting a remote *DomainParticipant* that resulted in a deadlock. These conditions were unexpected and were accompanied by exception log messages.

This issue has been resolved. If any of these conditions are hit, error messages are printed, but there is no longer risk of a deadlock.

[RTI Issue ID CORE-10556]

5.2.17 Samples not replaced when using Keep Last, Best Effort, finite max_samples, keyed data, and batching

Consider a scenario using DDS_BEST_EFFORT_RELIABILITY_QOS, DDS_KEEP_LAST_HISTORY_QOS, **max_samples**, keyed data, and batching. When using KEEP_LAST, a batch should never be dropped as long as it doesn't contain more samples than **max_samples/max_samples_per_remote_writer**. *Connext DDS* should replace samples that are currently in the queue with the samples in the batch.

However, when using DDS_BEST_EFFORT_RELIABILITY_QOS, *Connext DDS* rejected a batch when **max_samples** was hit instead of making space in the queue, due to DDS_KEEP_LAST_HISTORY_QOS replacement.

This scenario is now fixed. When a batch hits **max_samples**, *Connext DDS* makes space in the queue due to DDS_KEEP_LAST_HISTORY_QOS replacement.

[RTI Issue ID CORE-10580]

5.2.18 NOT_ALIVE_DISPOSED instances not transitioning to NOT_ALIVE_NO_WRITERS when using propagate_unregister_of_disposed_instances

When using **propagate_unregister_of_disposed_instances**, instances in the DDS_NOT_ALIVE_DISPOSED_INSTANCE_STATE did not transition to DDS_NOT_ALIVE_NO_WRITERS_INSTANCE_STATE if the transition was triggered by a *DataWriter* losing liveliness or being destroyed. This error has been fixed. Now instances transition to DDS_NOT_ALIVE_NO_WRITERS_INSTANCE_STATE in these cases.

[RTI Issue ID CORE-10663]

5.2.19 Unexpected errors while removing an instance from a DataWriter

A *DataWriter* in which the **on_instance_replaced** callback is installed may have generated the following errors when trying to remove an instance:

```
PRESWriterHistoryDriver_onReplaceInstance:!onInstanceReplaced
PRESPsService_writerHistoryDriverListenerOnInstanceReplaced:!modify pres psWriter
WriterHistoryMemoryPlugin_dropFullyAckedDisposedInstances:!on replace instance
WriterHistoryMemoryPlugin_applyFiniteAutopurgeDelay:!auto purge instance
```

These errors stopped the instance from being removed.

Now, the instance will be removed even if the errors appear.

[RTI Issue ID CORE-10696]

5.2.20 Loading Dbghelp.dll and NtDll.dll may have caused warnings

When the libraries Dbghelp.dll and NtDll.dll were loaded, the following warnings may have been logged:

```
RTIOsapiLibrary_getFullSharedLibraryName:library Dbghelp.dll, extension specified by user.
Consider removing extension to prevent library mismatches (release vs. debug)
RTIOsapiLibrary_getFullSharedLibraryName:library NtDll.dll, extension specified by user.
Consider removing extension to prevent library mismatches (release vs. debug)
```

These warnings are not logged anymore.

[RTI Issue ID CORE-10704]

5.2.21 Re-registering an instance did not restore correct state

Consider an instance that was disposed, then unregistered, and now you want to re-register it. Re-registering in this case transitioned the instance state to alive instead of disposed. This problem has been resolved. Re-registering a previously unregistered instance now restores the instance state to what it was before the unregister operation.

[RTI Issue ID CORE-10763]

5.2.22 Logging APIs did not configure verbosity of some Core Libraries log messages

Changing the logging verbosity did not affect the logging of some messages related to XML parsing, ODBC Dynamic Library Driver, and Property QoS policies. This issue has been resolved.

[RTI Issue ID CORE-10980]

5.2.23 Incorrect heap snapshot Information in some cases

When using the Heap Monitoring utility, some of the reported allocations were incorrectly categorized with an incorrect activity **PRESPsReaderQueue_newAnonData**.

If you were using the Heap Monitoring utility in previous versions and see allocations with this activity, these lines must be ignored in any analysis.

In this release, this issue has been fixed. Any allocations with that activity are now correct and should not be ignored.

[RTI Issue ID CORE-11094]

5.2.24 Memory leak and 'Inconsistent free/alloc and realloc/alloc' errors when using Heap Monitoring

When using Heap Monitoring, the following errors appeared:

```
heap.c:1011:inconsistent free/alloc: block id 0X37B1C10 being freed with "RTIOsapiHeap_
freeBufferNotAligned" and was allocated with "RTIOsapiHeap_allocateString"
RTIOsapiHeap_realloc:inconsistent realloc/alloc: block id 0XE22BE20 being realloced with
"RTIOsapiHeap_malloc" and was allocated with "RTIOsapiHeap_allocateString
```

When these errors occurred, *Connext DDS* leaked memory. These leaks happened only when Heap Monitoring was enabled.

In this release, *Connext DDS* no longer checks for inconsistency between realloc/alloc/free signatures when using Heap Monitoring. Now when you use Heap Monitoring, you will neither see these error messages nor experience memory leaks previously associated with them.

[RTI Issue ID CORE-11210]

5.2.25 RTI DDS Ping and RTI DDS Spy did not report error if QoS profile not found

If you passed an incorrect QoS profile name as an argument to *RTI DDS Ping* (**rtiddsping**) or *RTI DDS Spy* (**rtiddsspy**), these utilities did not report the problem and used the default QoS profile.

This problem has been resolved. Now an error will be logged that the QoS profile was not found and the default QoS profile will be used. For example, if you misspelled "Default" as "Defult", you may see a message such as this:

```
rtiddsping -qosFile TEST_QOS_PROFILE.xml -qosProfile myApp_Library::Defult_Profile
QoS profile 'myApp_Library::Defult_Profile' was not found.
Using default configuration.
```

[RTI Issue ID CORE-1145]

5.2.26 Memory leak in RTI DDS Ping and RTI Prototyper

There was a memory leak in *RTI DDS Ping* (**rtiddsping**) and *RTI Prototyper* (**rtiddsprototyper**):

```
==28275== 32 bytes in 1 blocks are still reachable in loss record 1 of 2
==28275==    at 0x4C2E216: operator new(unsigned long) (vg_replace_malloc.c:334)
==28275==    by 0x4F88E49: NDDSConfigLogger::get_instance() (Logger.cxx:52)
==28275==    by 0x40E7D8: NddsAgent::execute(char const*) (Agent.cxx:1166)
==28275==    by 0x40E78E: NddsAgent::execute(int, char const**) (Agent.cxx:1141)
```

This memory leak has been fixed.

[RTI Issue ID CORE-11151]

5.3 Fixes Related to Transports

5.3.1 Unexpected "MIGGenerator_addData:serialize buffer too small" error message

This issue was resolved in 6.0.1, but not documented at that time.

A *DataWriter* may have printed the following unexpected error message when the transports in the *DataWriter's* Participant were not configured with the same **message_size_max**:

```
MIGGenerator_addData:serialize
```

This problem only occurred when the *DataWriter* was sending data to best-effort DataReaders and may have caused samples to not be sent.

This problem has been resolved.

[RTI Issue ID CORE-2803]

5.3.2 Hostname resolution error messages printed regularly

Connext DDS printed error messages when trying to resolve a hostname that was unknown to the DNS Service. If the DNS Tracker was enabled, the result was that the error messages were printed regularly every time the DNS Tracker checked that hostname. This issue has been fixed. Now warning messages

are printed instead of errors. *Connext DDS* prints an error message only if not being able to resolve a host-name results in a later error.

[RTI Issue ID CORE-9840]

5.3.3 Network interface change not applied if change occurred while enabling DomainParticipant

If a change on the network interfaces happened while the *DomainParticipant* was being enabled, the change may have been discarded. This resulted in the *DomainParticipant* announcing incorrect locators until another change on the network interfaces happened. This issue has been fixed. Now the locators are updated properly.

[RTI Issue ID CORE-9922]

5.3.4 Still reachable memory leaks: TransportMulticastMapping libraries were never unloaded

If you specified any mapping functions and their libraries in the TransportMulticast QosPolicy, those libraries were loaded but never unloaded. This problem has been fixed by unloading the libraries after they are used during *DomainParticipant* creation and *DataReader* creation.

> **Note:** You may still see "still reachable" memory leaks in "dlopen" and "dlclose". These leaks are a result of a bug in Valgrind™ (https://bugs.launchpad.net/ubuntu/+source/valgrind/+bug/1160352).

[RTI Issue ID CORE-9941]

5.3.5 Deserialization error with BEST_EFFORT multicast readers when type was annotated for Zero Copy transfer over shared memory

Multicast *DataReaders* using a type annotated for Zero Copy transfer over shared memory did not receive samples due to a deserialization error with BEST_EFFORT reliability. This issue has been fixed.

[RTI Issue ID CORE-10083]

5.3.6 Possible bus error with shared memory transport on QNX or LynxOS platforms

When using the shared memory transport and rapidly creating and deleting *DomainParticipants*, it was extremely rare but possible for a separate *DomainParticipant* to encounter a bus error in the function **RTIOsapiSharedMemorySegment_attach_os()** while trying to send packets to those *DomainParticipants*. This problem, which only affected QNX and LynxOS platforms, has been fixed.

[RTI Issue ID CORE-10348]

5.3.7 Unexpected property: dds.transport.lbrtps.parent.domain_participant_ptr

If you were creating a *DomainParticipant* using the LBRTPS transport, you may have received the following error:

```
DDS_DomainParticipantConfigurator_setup_custom_transports:!create custom transport plugin
DDS_DomainParticipantConfigurator_enable:!install transport plugin aliases = custom transports
DDS_DomainParticipant_enableI:!enable transport configurator
DDS_DomainParticipantFactory_create_participant:ERROR: Failed to auto-enable entity
```

This problem has been fixed. Now you can create a participant using the LBRTPS transport, and the error message will not be logged.

[RTI Issue ID CORE-10409]

5.3.8 Precondition error when UDP debugging enabled in shared memory

If using debug libraries and **enable_udp_debugging** was set to "true", then the following error would occur in the internal function **NDDS_Transport_UDP_send** when sending an RTPS message with many repair samples:

```
!precondition: "self == ((void *)0) || buffer_in == ((void *)0) || buffer_count_in <= 0 ||
buffer_count_in > self->property->gather_send_buffer_count_max || worker == ((void *)0) ||
sendresource_in == ((void *)0) || *sendresource_in == ((void *)0)"
```

If this error occurred, the message would still be sent over shared memory, but it wouldn't be sent over UDP for debugging purposes. This problem has been fixed. The error and the UDP send failure no longer occur. If **enable_udp_debugging** is set to "true", then the number of shared memory transport gather buffers is now equal to the value of **parent.gather_send_buffer_count_max** or 16, whichever is smaller.

[RTI Issue ID CORE-10589]

5.3.9 Communication may have stopped working after an increase in the number of interfaces available in a host

Communication may have stopped working for a *DomainParticipant* that was running in a host for which the number of available interfaces increased. Specifically, if the number of available interfaces in a host increased, and if all of the new interfaces were malfunctioning, communication may have stopped even if the already existing available interfaces were still working fine.

This problem has been resolved: an increase in the interfaces available on a host should not result in communication issues.

[RTI Issue ID CORE-10611]

5.3.10 UDP properties_bitmap now supports string constant

Previously, the properties **dds.transport.UDPv4.builtin.properties_bitmap** and **dds.transport.UDPv6.builtin.properties_bitmap** only accepted numeric values.

This limitation has been resolved, and now those properties accept string constants too. **dds.transport.UDPv4.builtin.properties_bitmap**, **dds.transport.UDPv4_WAN.builtin.properties_bitmap**, and **dds.transport.UDPv6.builtin.properties_bitmap** support:

```
"1",
"2",
"TRANSPORT_PROPERTY_BIT_BUFFER_ALWAYS_LOANED",
"NDDS_TRANSPORT_PROPERTY_BIT_BUFFER_ALWAYS_LOANED",
"TRANSPORT_PROPERTY_BITMAP_DEFAULT",
"NDDS_TRANSPORT_PROPERTIES_BITMAP_DEFAULT"
```

[RTI Issue ID CORE-10989]

5.3.11 TCP transport could not parse gather_send_buffer_count_max property

The TCP transport plugin was unable to parse the property **dds.transport.TCPv4.tcp1.parent.gather_send_buffer_count_max**. Therefore you could not improve the performance of the write operation by optimizing the number of buffers that *Connext DDS* can pass to the **send()** method of a transport plugin. This problem has been resolved.

[RTI Issue ID COREPLG-544]

5.3.12 Memory leak in debug logging for TCP transport

The combination of debug libraries and the verbosity RTI_LOG_BIT_OTHER in TCP Transport had a memory leak:

```
==5542==
==5542== HEAP SUMMARY:
==5542==     in use at exit: 56 bytes in 1 blocks
==5542==   total heap usage: 1,030 allocs, 1,029 frees, 3,158,200 bytes allocated
==5542==
==5542== 56 bytes in 1 blocks are still reachable in loss record 1 of 1
==5542==    at 0x4C2FB55: calloc (in /usr/lib/valgrind/vgpreload_memcheck-amd64-linux.so)
==5542==    by 0xD7C6C8: RTIOsapiHeap_reallocateMemoryInternal (heap.c:771)
==5542==    by 0xD7405D: RTISystemClock_new (SystemClock.c:392)
==5542==    by 0x446183: NDDS_Transport_TCPv4_logDebug (Tcpv4.c:984)
==5542==    by 0x46A2C4: NDDS_Transport_TCPv4_new (Tcpv4.c:12559)
==5542==    by 0x42DD14: NDDS_Transport_TCPv4Tester_testLoggingVerbosityWithParams
(Tcpv4Tester.c:4257)
==5542==    by 0x42E374: NDDS_Transport_TCPv4Tester_testLoggingVerbosity (Tcpv4Tester.c:4341)
==5542==    by 0x47E86B: RTITestSetting_runTestsExt (Setting.c:893)
==5542==    by 0x47F81B: RTITestSetting_runTests (Setting.c:1089)
==5542==    by 0x42E560: NDDS_Transport_TCPv4Tester_run (Tcpv4Tester.c:4421)
```

```
==5542==    by 0x47E86B: RTITestSetting_runTestsExt (Setting.c:893)
==5542==    by 0x47F81B: RTITestSetting_runTests (Setting.c:1089)
==5542==
==5542== LEAK SUMMARY:
==5542==    definitely lost: 0 bytes in 0 blocks
==5542==    indirectly lost: 0 bytes in 0 blocks
==5542==      possibly lost: 0 bytes in 0 blocks
==5542==    still reachable: 56 bytes in 1 blocks
==5542==         suppressed: 0 bytes in 0 blocks
==5542==
==5542== For counts of detected and suppressed errors, rerun with: -v
==5542== ERROR SUMMARY: 0 errors from 0 contexts (suppressed: 0 from 0)
```

This problem has been resolved by cleaning the memory in the destructor of the TCP Transport plugin.

[RTI Issue ID COREPLG-520]

5.3.13 TCP Transport did not close sockets upon shutdown

An issue may have provoked the TCP Transport to not properly close one or more sockets. The issue may have been triggered during TCP Transport shutdown. When it happened, the following unexpected message was logged at the EXCEPTION log level:

```
NDDS_Transport_TCPv4_delete_cEA:unexpected situation: got already destroyed connection
```

This issue is resolved. Now the TCP Transport should not log the unexpected message nor leak any socket.

[RTI Issue ID COREPLG-545]

5.4 Fixes Related to Reliability Protocol and Wire Representation

5.4.1 Memory leak when failing to create a reliable DataWriter due to port collision

A reliable *DataWriter* requires a port for receiving ACKNACKs from reliable *DataReaders*. The *DataWriter's* TransportUnicast QosPolicy determines the port number. If the port is already in use, the *DataWriter* creation will fail. This failure scenario resulted in a memory leak in the function **COMMENDSrWriterService_createWriter**. This memory leak has been fixed.

[RTI Issue ID CORE-9775]

5.4.2 Unnecessary periodic heartbeats sent when writer had never written any samples

A *DataWriter* that had never written any samples still sent periodic heartbeat messages announcing the empty queue and generating unnecessary network traffic.

This issue has been resolved. A writer will not start sending periodic heartbeats until it has written its first sample.

[RTI Issue ID CORE-9795]

5.4.3 Excess samples NACKed by DataReaders in rare situations

In rare situations, a *DataReader* may have NACKed more samples than could fit in its queue, causing rejection and further NACKing to occur. This situation could happen when the *DataReader* was matched with two *DataWriters* with identical virtual identities (e.g., a system with redundant Routing Services). If the *DataReader* application was run with debug libraries, the following precondition error was printed in this situation:

```
MIGRtpsBitmap_truncateToZeroCount:!precondition: me == ((void *)0) || startSn == 0 ||
totalZeros < 0
```

This issue has been resolved.

[RTI Issue ID CORE-9864]

5.4.4 Unexpected "WriterHistoryMemoryPlugin_ removeRemoteReader:!change app ack state" error when using AppAck on a DataReader whose participant lost liveliness

When a *DataReader* was configured to use application-level acknowledgement, and its *DomainParticipant* lost liveliness, you may have seen the following error in the *DataWriter's* application:

```
"WriterHistoryMemoryPlugin_removeRemoteReader:!change app ack state"
```

The error occurred if:

- The *DataWriter's* application was installing the **on_sample_removed** callback.
- The *DataReader* received the samples from the *DataWriter* using Zero Copy transfer over shared memory. Also, in this scenario, the *DataWriter* may eventually not have been able to continue writing samples.

This problem has been resolved.

[RTI Issue ID CORE-9985]

5.4.5 Wrong memory allocation when deserializing an unbounded (w)string with a wrong length

If the serialized length of an unbounded string or wstring was corrupted over the network, then the receiver may have incorrectly attempted to allocate an amount of memory equal to this corrupted length. If the corrupted length was large enough, certain architectures may have crashed during this attempt. This

problem has been fixed by checking the size against the remaining length of the DATA submessage before allocating memory.

[RTI Issue ID CORE-10059]

5.4.6 Protocol status by locator may have been wrong with reliable multicast communications

Calling **DataWriter::get_matched_subscription_datawriter_protocol_status_by_locator** for a reliable *DataWriter* matching with a reliable *DataReader* that configures multicast may have returned incorrect results if the *DataWriter* sent repair data to the *DataReader* using the *DataReader's* unicast locators.

In this case, the *DataWriter* updated the repair traffic protocol statistics for the multicast locator instead of the unicast locator.

This problem has been resolved.

[RTI Issue ID CORE-10221]

5.4.7 max_bytes_per_nack_response not used correctly with ASYNCHRONOUS_PUBLISH_MODE_QOS

When a reliable *DataWriter* resends DDS samples, the maximum size of a NACK repair packet is limited to the **max_bytes_per_nack_response** value. To improve bandwidth utilization and response latency, a *DataWriter* tries to use the whole **max_bytes_per_nack_response** if possible.

For example, if a *DataReader* NACKs 4 samples with serialized sizes 4, 10, 10, 10 and **max_bytes_per_nack_response** is 30, the *DataWriter* will send the first 3 samples into the repair packet. Sending the fourth sample would exceed **max_bytes_per_nack_response**.

Previously, when the *DataWriter* was configured to use ASYNCHRONOUS_PUBLISH_MODE_QOS, the *DataWriter* may not have utilized the full **max_bytes_per_nack_response** and the repair packet may have been smaller than expected. (Using the example here, the NACK response may have included only the first sample with size 4 instead of the first 3 samples.) This problem occurred only when the NACK response was being sent to multiple locators.

This problem has been resolved.

[RTI Issue ID CORE-10459]

5.4.8 Inefficient delivery of samples with reliable asynchronous publisher

DataWriters using asynchronous publishing may have sent samples inefficiently when live samples and repair samples were being sent at the same time by the asynchronous publishing thread. In some cases, live samples may have been inadvertently directed to only a single *DataReader* rather than to all matching

DataReaders. The effect of this was that the other *DataReaders* had to NACK for those live samples and get them repaired separately, increasing latency and bandwidth usage.

This issue has been resolved.

[RTI Issue ID CORE-10495]

5.4.9 Samples may not have been automatically acknowledged on a DataWriter when a DataReader using application-level acknowledgment was deleted or lost liveliness

Some samples may not have been automatically acknowledged on a *DataWriter* when a *DataReader* using application-level acknowledgement was deleted or lost liveliness. As a result, these samples may have never been removed from the *DataWriter* queue, leading to potential resource exhaustion.

This issue occurred only for *DataReaders* that did not acknowledge any samples implicitly (by reading or taking samples from the reader queue) or explicitly (by using the **DataReader::acknowledge** APIs) before they were deleted.

This problem has been resolved.

[RTI Issue ID CORE-10682]

5.5 Fixes Related to Content Filters and Query Conditions

5.5.1 Duplicate samples sent unnecessarily to DataReaders within the same DomainParticipant when using ContentFilteredTopics

A *DataWriter* sent a sample to each *DataReader* within a *DomainParticipant* for which the sample passed the content filter. However, sending a single copy of the sample was sufficient because the *DataReaders* all processed the sample the first time it was received and dropped the subsequent copies. This issue has been fixed. Now when a sample should be delivered to multiple *DataReaders* within a *DomainParticipant*, it is sent only a single time.

[RTI Issue ID CORE-8993]

5.5.2 ContentFilteredTopic performance improvement

An unnecessary buffer initialization in the code that filters data samples for a ContentFilteredTopic or a *QueryCondition* has been removed and will result in faster filtering, especially for large data samples.

[RTI Issue ID CORE-10116]

5.5.3 Reader-side filtering did not work with Zero Copy transfer over shared memory

Performing content filtering (via a ContentFilteredTopic) on the *DataReader* side was not possible when using Zero Copy transfer over shared memory. (See the section "5.4.2 Where Filtering is Applied---Publishing vs. Subscribing Side," in the *RTI Connext DDS Core Libraries User's Manual*. See also "23.6 Zero Copy Transfer Over Shared Memory.")

For *DataReader*-side filtering, *Connext DDS* filters samples at reception time, but for Zero Copy samples, the *DataWriter* sends an associated reference to the actual sample, so the *DataReader* could not filter this reference.

This problem has been fixed. Now, for Zero Copy samples, the filtering operation is delayed until the *DataReader* has access to the shared memory sample. As a result, samples sent via Zero Copy transfer over shared memory are now properly filtered on the *DataReader* side. (This problem only occurred when filtering was performed by the *DataReader*.)

[RTI Issue ID CORE-10118]

5.5.4 ContentFilteredTopic::append/remove_from_expression_parameter crashed when bad index was passed

These functions crashed when a negative index or an index equal to the parameter length was passed. This problem has been fixed. The operation now fails with DDS_RETCODE_BAD_PARAMETER (or the equivalent exception).

[RTI Issue ID CORE-10298]

5.5.5 DDS_DomainParticipant_create_contentfilteredtopic_w_filter: possible crash with string-match filter

The function **DDS_DomainParticipant_create_contentfilteredtopic_w_filter()** may have crashed under the following conditions:

- The filter name was DDS_STRINGMATCHFILTER_NAME, and
- A nonempty sequence of parameters that were NOT allocated with a **DDS_String_*** function was used.

This function violated its contract and could modify (reallocate) the strings in the input parameter sequence.

The default filter (DDS_SQLFILTER_NAME) was not affected by this problem.

This problem has been resolved. Now this function never modifies its input parameters.

[RTI Issue ID CORE-10299]

5.5.6 GAPs from ContentFilteredTopic were counted incorrectly in max_bytes_per_nack_response

When a reliable *DataWriter* resends DDS samples, the repair packet size is limited to the **max_bytes_per_nack_response** value. Previously, when computing **max_bytes_per_nack_response**, samples that were filtered out were counted according to their serialized size, rather than to their corresponding GAP size. This problem was not observed while using asynchronous *DataWriters*.

This problem has been resolved. Now samples that are filtered out are counted according to their GAP size.

[RTI Issue ID CORE-10335]

5.5.7 WaitSet with QueryCondition/ReadCondition may not have woken up when entities changed to not compatible or were removed

In a scenario with multiple local *DataReaders* matching a remote *DataWriter*, where one of the following two events occurs:

- The remote writer was no longer compatible with the local readers.
- The remote writer was removed.

If you were using a WaitSet with QueryCondition or ReadCondition, you may not have been notified by the QueryCondition/ReadCondition because the WaitSet may not have woken up. So you may not have received an invalid sample with the instance_state: DDS_NOT_ALIVE_NO_WRITERS_INSTANCE_STATE.

This issue has been resolved. Now the WaitSet will wake up in the above described scenarios.

[RTI Issue ID CORE-10365]

5.5.8 Invalid QueryCondition and ReadCondition results for samples that expired due to Lifespan QoS while loaned

QueryCondition or ReadCondition results may have been invalid if an application held a loan on a sample that expired due to lifespan during the time that the sample was loaned. This could have led to situations where WaitSets returned indicating that there were active conditions when there were not, or one of the **read/take_w_condition** APIs returned indicating that there was data available when there was not.

This issue has been resolved. When a sample's lifespan expires while it is loaned, it is correctly removed from all existing QueryConditions and ReadConditions as soon as the outstanding loan is returned.

[RTI Issue ID CORE-10746]

5.6 Fixes Related to TopicQueries

5.6.1 MultiChannel and TopicQuery did not work with large data

A *DataReader* was sometimes not able to receive large data (samples bigger than the transport **message_size_max**) from a MultiChannel *DataWriter*, of which a TopicQuery dispatcher is an example. This problem occurred because the original identity of a sample was not preserved when sending multi-channel large data. This issue and CORE-8422, which was incorrectly marked as fixed in 5.3.1, are now resolved.

[RTI Issue ID CORE-9335]

5.6.2 create_topic_query hanged when setting service_request_writer_data_lifecycle

Setting the DiscoveryConfig QosPolicy's **service_request_writer_data_lifecycle** to have a finite value for **autopurge_unregistered_instances_delay** or **autopurge_disposed_instances_delay** resulted in incorrect behavior when repeatedly creating and deleting topic queries. With release libraries, **DDS_DataReader_create_topic_query** would hang. With debug libraries, you would see the following errors every time the ServiceRequest *DataWriter* sent a heartbeat:

```
REDACursor_start:!precondition: !( ((c)!=((void *)0)) && !(((c)->_state) & 0x02) )
PRESPsService_writerHistoryDriverListenerOnInstanceReplaced:!start pres psWriter
PRESWriterHistoryDriver_onReplaceInstance:!onInstanceReplaced
```

This precondition error occurred for any *DataWriter* that instrumented the **on_instance_replaced** listener callback and set a finite value for a WriterDataLifecycle QosPolicy duration. This problem has been fixed.

[RTI Issue ID CORE-10046]

5.6.3 Historical TopicQueries and ContentFilteredTopics may have been out of synch

Some use cases require that a *DataReader* requests all historical data matching a given filter expression in addition to subscribing to all live data matching this filter expression. The historical data can be requested using an historical TopicQuery and the live data can be filtered using a ContentFilteredTopic expression. Doing so allows the *DataReader* a continuous view of a data stream from past to present.

There was an issue, however, that may have caused a gap in the data in between the historical and live data streams. This would happen when the TopicQuery was received by the *DataWriter* before the matching ContentFilteredTopic filter expression update was received.

This issue has been resolved. TopicQueries and ContentFilteredTopic filter expression updates are now synchronized on the *DataWriter*. That said, as a matter of best practice, you should use continuous TopicQueries instead of using both historical TopicQueries and ContentFilteredTopics in order to address the use case described here.

[RTI Issue ID CORE-10146]

5.6.4 Unregistered samples for TopicQueries may have been delivered even after using "@instance_state = ALIVE" in filter expression

The "@instance_state = ALIVE" modifier to filter expressions in TopicQueries only considered disposed samples. Unregistered samples were still sent by the *DataWriter*. This problem has been fixed.

[RTI Issue ID CORE-10604]

5.6.5 Unexpected "topic query does not exist" messages at warning level

While using TopicQueries, you may have seen the following log message at a warning level:

```
PRESPsService_removeRemoteTopicQuery:topic query does not exist
```

These messages are expected when TopicQuery requests are received out of order; they should not be reported at a warning verbosity level.

This issue has been fixed.

[RTI Issue ID CORE-10605]

5.6.6 Crash when TopicQuery could not be enabled

If a TopicQuery failed to be enabled, the application that was trying to create the TopicQuery would crash.

TopicQueries are enabled in the context of the **DataReader::create_topic_query()** call if the *DataReader* is enabled; otherwise, they are enabled at a later point during the call to enable the DataReader.

If a TopicQuery failed to be enabled, the following (or similar) errors would be printed:

```
DDS_DataReader_enable_topic_queryI:!announce TopicQuery
PRESPsService_enableTopicQueryWithCursor:!enable listener notification
PRESTopicQuery_enable:!enable topic query
DDS_TopicQuery_enable:!enable TopicQuery
DDS_DataReader_create_topic_queryI:!enable TopicQuery
```

This issue has been resolved. Now, if a TopicQuery cannot be enabled, there will still be errors printed indicating that there was an error, but no crash.

[RTI Issue ID CORE-11295]

5.7 Fixes Related to Coherent Sets

5.7.1 Unhandled exception when copying SampleInfo and accessing SampleInfo.coherent_set_info field

In release 6.0.0.6, copying a SampleInfo object where the field **coherent_set_info** is set and accessing the **coherent_set_info** field in the copied object may have thrown an unhandled exception.

This issue has been resolved.

Note: This issue affected only releases 6.0.0.6 and 6.0.0.11 because it affected a feature that is not part of the 6.0.0 and 6.0.1 releases.

[RTI Issue ID CORE-10408]

5.7.2 Unexpected DDS_RETCODE_ERROR when writing a sample with durable writer history

The **DataWriter::write()** operation may have failed with DDS_RETCODE_ERROR, entering a non-recoverable state. This was due to a corruption in a sample metadata list maintained in the *DataWriter*, when the *Publisher* called **end_coherent_changes()** with a previously published coherent set in the *DataWriter* history. This could also have led to the *DataReader's* losing samples. This error typically occurred with the following configuration:

- Communication was reliable.
- Durable writer history was set (see the "Durable Writer History Properties" table in the *RTI Connext DDS Core Libraries User's Manual*), with the property **dds.data_writer.history.odbc_plugin.in_memory_state** set to true.
- Coherent sets of samples were published.
- The *DataWriter* set **max_samples** to a finite value.

This issue has been resolved.

[RTI Issue ID CORE-10498]

5.7.3 SampleInfo.equals in Java may have returned false negatives

Calling **SampleInfo.equals** may have returned false when comparing two **SampleInfos** that were equal.

This issue occurred only when the **coherent_set_info** field was set. This problem has been resolved.

[RTI Issue ID CORE-10600]

5.7.4 Segmentation fault when using coherent sets on keyed Topics

A subscribing application linking with the *Connext DDS* release libraries may have experienced a segmentation fault when receiving a coherent set containing samples from multiple instances. The issue only occurred when an instance contained more than one sample in the coherent set.

If the application was linked with the debug libraries, you would have observed the following precondition error in an infinite loop:

```
!precondition: "instanceEntry == ((void*)0)"
```

This problem has been resolved.

[RTI Issue ID CORE-11225]

5.7.5 Coherent set may not have been delivered atomically

It was possible that a coherent set was not delivered atomically to the subscribing application. This meant that you could get some samples of the coherent set first and later on the rest. This problem only occurred when **max_samples_per_instance** was set to a finite number on the *DataReader* QoS.

This problem has been resolved.

[RTI Issue ID CORE-11227]

5.8 Fixes Related to Dynamic Data and FlatData

5.8.1 FlatData: plain_cast may have incorrectly allowed access to memory that was not properly aligned in some situations

The function **rti::flat::plain_cast()** allowed casting sequences and arrays of fixed-size structs from their FlatData representation to a C++ array even though the memory alignment wasn't C++-compatible.

This problem has been resolved. **plain_cast** will now fail in these situations instead of providing a pointer to a misaligned array.

[RTI Issue ID CORE-10093]

5.8.2 Using DynamicData::get_complex_member or DynamicData::set_complex_member on a type that contains sequences of strings or wide strings could have led to sample corruption or segmentation fault

Using the **DynamicData::get_complex_member** or **DynamicData::set_complex_member** APIs to get or set a member with a type that contained a sequence of strings or a sequence of wide strings could have led to sample corruption or a segmentation fault.

This issue has been fixed.

[RTI Issue ID CORE-11187]

5.9 Fixes Related to DDS API

5.9.1 DataReader::get_matched_publications may not have returned all the matched DataWriter handles when using MultiChannel

In previous releases, there was an issue with the **DataReader::get_matched_publications** API when using MultiChannel that may have resulted in the API returning an incomplete set of matched *DataWriter* handles. This problem only occurred when the *DataReader* changed its filter expression and that change resulted in matching a different set of channels for a given *DataWriter*.

This problem is now resolved: **DataReader::get_matched_publications** will always return the correct set of matched *DataWriter* handles.

[RTI Issue ID CORE-6944]

5.9.2 Wrong return code or exception for DDS_DataWriter_get_matched_subscription_data and DDS_DataReader_get_matched_publication_data

DDS_DataWriter_get_matched_subscription_data and **DDS_DataReader_get_matched_publication_data** incorrectly returned DDS_RETCODE_PRECONDITION_NOT_MET instead of DDS_RETCODE_BAD_PARAMETER when the instance handle did not correspond to any matched endpoint. This problem also affected APIs that use exceptions instead of return codes (Modern C++, Java, and .NET). This problem has been fixed.

[RTI Issue ID CORE-10103]

5.9.3 Unexpected log message when calling DataWriter::get_matched_subscription_data or DataReader::get_matched_publication_data on unmatched endpoints

The **DataWriter::get_matched_subscription_data** and **DataReader::get_matched_publication_data** APIs return RETCODE_PRECONDITION_NOT_MET when called using subscription or publication handles of endpoints that do not match with the calling endpoint. This is normal operation for the API and should not produce any logging messages at the exception log level; however, starting in release 6.0.0, an exception was printed in this case. This issue has been fixed. The log message is now printed at the warning log level, as was the case in releases previous to 6.0.0.

[RTI Issue ID CORE-10163]

5.9.4 FooDataReader::get_key_value() may have returned wrong key value

Calling **FooDataReader::get_key_value()** may have returned a wrong key value. This occurred for types containing mutable non-primitive key members and only if the non-primitive type of a key member did not contain any key itself. For example:

```
@mutable
struct Identifier {
    int32 x;
    int32 y;
};
@mutable
struct Entity {
    @key Identifier ID;
    int32 other;
};
```

This problem has been resolved.

[RTI Issue ID CORE-10884]

5.9.5 DDS_WaitSetProperty::max_event_count incorrectly declared as long (C and Traditional C++ APIs only)

The **max_event_count** field of the type DDS_WaitSetProperty was incorrectly declared as long (or int32) in the C and Traditional C++ APIs. This field was always intended to be a 32-bit integer, but C's int32 type size is not standard. Depending on the architecture, int32 can be a 32-bit or a 64-bit integer.

This problem has been resolved, and **max_event_count**'s type is now DDS_Long (which is always a 32-bit integer).

[RTI Issue ID CORE-10965]

5.9.6 Crash when calling NDDSConfigLogger::finalize_instance() twice

When calling **NDDSConfigLogger::finalize_instance()** twice, there was a crash with the following stack trace:

```
#0 0x00007f6249b9f01a in NDDSConfigLogger::set_output_device (this=0x0, device=0x0)
at Logger.cxx:138
#1 0x00007f6249b9ee19 in NDDSConfigLogger::finalize_instance () at Logger.cxx:59
```

This issue has been fixed. Now you can call **NDDSConfigLogger::finalize_instance()** more than once.

[RTI Issue ID CORE-11134]

5.10 Fixes Related to Modern C++ API

In addition to 5.9 Fixes Related to DDS API on page 57, this release includes the following fixes, which are specific to the Modern C++ API.

5.10.1 Incorrect call to write method with TopicInstance types

Using TopicInstance iterators to write resulted in a compilation error. Now, the way TopicInstance types are handled in the corresponding call has been fixed to perform the write operation correctly.

[RTI Issue ID CORE-9988]

5.10.2 Non-uniform naming for data_tag

Some parts of the module referred to data_tag as data_tags, making them sometimes incompatible. References are now uniform across the API.

[RTI Issue ID CORE-9991]

5.10.3 Some types had copy constructor but no explicit assignment operator

Some types, such as Time, all the Exception types, and InstanceHandle had a copy constructor but no explicit assignment operator. This may have caused some static-code-analysis tools to report errors.

In all cases, the compiler-defined operator was correct, so this problem didn't have any functional effect.

In cases where the copy constructor wasn't necessary (the compiler-defined one was appropriate), the constructor has been removed; in the rest of cases, a copy-assignment operator has been added.

[RTI Issue ID CORE-10000]

5.10.4 Some headers were included recursively

Some headers were included recursively, which may have caused static-code-analysis tools to report an error. This problem didn't have any effect on the compilation or API functionality. This problem has been fixed.

[RTI Issue ID CORE-10001]

5.10.5 For DynamicData DataWriters, the {{create_data()}} member function didn't compile

Given a writer of type **dds::pub::DataWriter<dds::core::xtypes::DynamicData>**, the following didn't compile:

```
dds::core::xtypes::DynamicData sample = writer.extensions().create_data();
```

This problem didn't affect IDL-generated types.

The problem has been resolved. The above expression now compiles and creates a DynamicData sample for the DynamicType of the *DataWriter's* Topic.

[RTI Issue ID CORE-10025]

5.10.6 Function to get type definition of a registered type was missing

The following function, available in the other language APIs, was not available in the Modern C++ API. It has now been added to the **rti::domain namespace**:

```
dds::core::xtypes::DynamicType& find_type(
const dds::domain::DomainParticipant& participant,
const std::string& type_name);
```

[RTI Issue ID CORE-10044]

5.10.7 Time::from_millisecs and Time::from_microsecs could produce incorrect results

These two methods could return an incorrect result when the input couldn't be represented as a 32-bit integer. This problem has been resolved. Now these functions return the right Time for all possible inputs.

[RTI Issue ID CORE-10100]

5.10.8 New method to configure the default QosProvider

The function to set the default QosProvider parameters, which configure among other things which QoS profile files are loaded by default, was not directly accessible without instantiating **QosProvider::Default ()** first. This could have undesired side effects.

A new, standalone function, **rti::core::default_qos_provider_params()**, has replaced the previous one. This function can be called before **QosProvider::Default()** is first accessed.

[RTI Issue ID CORE-10132]

5.10.9 Applications that used a StatusCondition from an XML-loaded DDS entity may have crashed in some situations

Applications that loaded an XML-defined DDS system (via **QosProvider::create_participant_from_ config**) may have crashed if they used the **StatusCondition::entity()** getter to get the entity related to a *StatusCondition*. The *StatusCondition* did not retain the reference to the entity, and in some situations the entity may have been destroyed, causing **StatusCondition::entity()** to return a dangling reference.

This problem has been resolved.

[RTI Issue ID CORE-10248]

5.10.10 Some DynamicData value setters and the member_info function may have incorrectly thrown an exception

Some DynamicData value setters (**value()** or **set_values()** for certain member types) may have incorrectly thrown an exception when accessing a member or element in the following situations:

- For a union type, when the member of interest wasn't currently selected by the discriminator. The expected behavior in this case is to automatically update the union discriminator, instead of failing.
- For a sequence type, when the element index was greater than the sequence length (but smaller than the maximum length). The expected behavior in this case is to automatically grow the sequence.
- When the member of interest is optional and currently unset. In this case, the expected behavior is to select the member.

In the case of **DynamicData::member_info()**, the expected behavior in the above scenarios is to return a **DynamicDataMemberInfo** with **member_exists** set to false, not to throw an exception.

This problem has been resolved. All the scenarios described above produce the expected behavior and no longer trigger an exception. These functions still throw an exception if the member doesn't exist in the type definition or if, in the case of sequences, the index is greater than the maximum length.

[RTI Issue ID CORE-10286]

5.10.11 Reference type had copy constructor but no explicit assignment operator

This problem may have caused some static-code-analysis tools to report errors. However, the compiler-defined operator is correct, so this problem didn't have any functional effect.

The explicit definition of the copy constructor wasn't necessary and has been removed, resolving this issue.

[RTI Issue ID CORE-10339]

5.10.12 Function rti::topic::find_topics not exported on Windows

Visual Studio applications using the function **rti::topic::find_topics** failed to link because that function wasn't correctly dll-exported.

This problem has been resolved.

[RTI Issue ID CORE-10636]

5.10.13 Some reference types didn't provide move constructors or move-assignment operators

Standard reference types such as DomainParticipant unnecessarily defined empty destructors, which disabled the default move constructor and move-assignment operators. The underlying **shared_ptr** in these types was therefore copied in situations where it could be moved, which could have been less efficient.

This problem has been resolved. Reference types are now nothrow move constructible and nothrow move assignable.

[RTI Issue ID CORE-10822]

5.11 Fixes Related to XML Configuration

5.11.1 Default QosProvider failed to apply certain Qos settings as defined in XML

A problem in the way the Default QosProvider loaded the XML Qos definitions caused a number of settings to not be applied, namely:

- The <participant_factory_qos> set in a <qos_profile> marked with the attribute **is_default_participant_factory_profile="true"** in a file other than USER_QOS_PROFILES.xml was not applied. For example, a profile changing the logging policy to write to a file would not be applied, and the application loading that profile would still print log messages on the console.
- When creating DDS Entities from an XML file (via **QosProvider::create_participant_from_config()**), those entities did not automatically get the Qos configuration from a profile marked with **is_default_qos=true** in that same file.

These problems are now resolved.

Note: This fix resolves a known issue in previous releases, "DomainParticipantFactoryQos in XML may not be Loaded."

[RTI Issue ID CORE-6846]

5.11.2 DomainParticipantFactory and QosProvider did not pick up the default XML QoS profile marked with is_default_qos

Previously, if an XML QoS profile had the **is_default_qos=true** attribute set, the DomainParticipantFactory or the QosProvider didn't update its default profile accordingly. This meant that the DomainParticipantFactory and QosProvider operations that expect a profile name as an argument didn't take into account the default one in the XML file when no profile argument was provided.

For example, given a file "my_profiles.xml" defining the following profile:

```
<dds>
    <qos_library name="test_library">
        <qos_profile name="test_profile" is_default_qos="true">
            <datareader_qos>
                <reliability>
                    <kind>RELIABLE_RELIABILITY_QOS</kind>
                </reliability>
            </datareader_qos>
        </qos_profile>
    </qos_library>
</dds>
```

The following C++ code didn't work as expected:

```
using namespace dds::all;
QosProvider qos_provider("my_profiles.xml");
auto qos = qos_provider.datareader_qos();
std::cout << (reader_qos.policy<Reliability>().kind() == ReliabilityKind::RELIABLE); //
expected true, but it's false
qos = qos_provider.datareader_qos("test_library::test_profile");
std::cout << (reader_qos.policy<Reliability>().kind() == ReliabilityKind::RELIABLE); // true
```

This problem has been resolved. Now in that example, the returned reader QoS has **ReliabilityKind::RELIABLE** in both cases.

[RTI Issue ID CORE-9497]

5.11.3 Could not configure force_interface_poll_detection and join_multicast_group_timeout using XML

The properties f**orce_interface_poll_detection** and **join_multicast_group_timeout** were not configurable in an XML QoS profile. This issue has been resolved. Now, these properties can be configured using the <udpv4> or <udpv6> tags in <transport_builtin>.

[RTI Issue ID CORE-9901]

5.11.4 Segmentation fault when loading invalid XML with invalid unions

Connext DDS may have produced a segmentation fault after trying to load an invalid XML file containing a malformed <union>. For example:

```
<union name="MyStruct">
<discriminator type="int32"/>
<case>
    <caseDiscriminator value="1"/>
    <member name="a" type="int32" />
    <member name="b" type="int64" />
    <member name="c" type="char" />
</case>
</union>
```

In the above XML snippet, a union has multiple members for a discriminator value, which is incorrect.

This problem has been resolved.

[RTI Issue ID CORE-10204]

5.11.5 XML fields ignore_enum_literal_names and initialize_writer_loaned_sample were not inherited

Starting in release 6.0.0, the fields **DataReaderQos::type_consistency::ignore_enum_literal_names** and **DataWriterQos::writer_resource_limits::initialize_writer_loaned_sample** were not inherited properly. This problem has been resolved.

[RTI Issue ID CORE-10214]

5.11.6 XSD validation failed if flags used a combination of values

The XSD validation of an XML application file failed if there was a UDPv4 configuration using a combination of values for the flags element. For example, using the following snippet in MAG reported the following error:

```
<transport_builtin>
    <udpv4>
        <interface_table>
            <element>
<flags>UDP_INTERFACE_INTERFACE_UP_FLAG|UDP_INTERFACE_INTERFACE_MULTICAST_FLAG</flags>
            </element>
        </interface_table>
    </udpv4>
</transport_builtin>
```

```
ERROR com.rti.micro.appgen.MicroAppGen - cvc-pattern-valid: Value 'UDP_INTERFACE_INTERFACE_UP_
FLAG|UDP_INTERFACE_INTERFACE_MULTICAST_FLAG' is not facet-valid with respect to pattern '(UDP_
INTERFACE_INTERFACE_UP_FLAG|UDP_INTERFACE_INTERFACE_MULTICAST_FLAG)' for type
'udpInterfaceFlagMask'.
```

This problem has been fixed. Now combinations are allowed.

[RTI Issue ID CORE-10314]

5.12 Fixes Related to OMG Specification Compliance

5.12.1 Connext DDS may have received wrong Simple Endpoint Discovery information when interoperating with other vendors

When a *Connext DDSDomainParticipant* received Simple Endpoint Discovery information from a different vendor participant, the received discovery information could be wrong. The issue may have occurred when the remote non-RTI participant did not serialize the vendorId field (which was an RTI-extension for Simple Endpoint Discovery) as part of the Simple Endpoint Discovery.

This issue has been resolved. If the vendorId is not in the Publication/SubscriptionBuiltinTopicData, it is now derived from the RTPS header.

This fix is part of the new functionality described in "Connext DDS now supports receiving implicit discovery information from the RTPS header" in RTI Connext DDS Core Libraries What's New in 6.1.1.

[RTI Issue ID CORE-9265]

5.12.2 Wrong default values in TypeConsistencyEnforcementQosPolicy

The TYPE_CONSISTENCY_ENFORCEMENT QosPolicy had wrong default values for two of its boolean members: **ignore_sequence_bounds** and **ignore_string_bounds**. This problem has been fixed.

To comply with the "Extensible and Dynamic Topic Types for DDS" specification from the Object Management Group (OMG), the default values have changed from **false** to **true**.

As a result of this change, out of the box in 6.1.0, a *DataWriter* publishing a *Topic* with a type that contains a sequence member with a maximum bound 'A' will match with a *DataReader* subscribing to the same *Topic* using a type where the same sequence member has a smaller maximum bound.

Samples received by the *DataReader* will fail to be deserialized and be dropped if the number of elements in the sequence is larger than the maximum allowed by the *DataReader* type.

For example:

```
struct MyType1 {
    sequence<int32, 100> m1;
}

struct MyType2 {
    sequence<int32, 50> m1;
}
```

In earlier releases, a *DataWriter* publishing **MyType1** and a *DataReader* subscribing to **MyType2** did not match out of the box. In 6.1.0, the entities will match.

If the *DataReader* receives a sample where **m1** has 8 elements, it will log a deserialization error and the sample will be reported as lost with the reason DDS_LOST_BY_DESERIALIZATION_FAILURE.

[RTI Issue ID CORE-9338]

5.12.3 APIs that provide information about remote entities were not compliant with specification

Previously, the **DDS_DataWriter_get_matched_subscriptions** and **DDS_DataReader_get_matched_publications** APIs only returned instance handles for remote entities that were alive. This was not compliant with the OMG Data Distribution Service (DDS) standard API, version 1.4. Now, these APIs return the instance handles for any matching remote entities, including those that are not alive.

Additionally, the **DDS_DataWriter_get_matched_subscription_data** and **DDS_DataReader_get_matched_publication_data** APIs used to accept any valid instance handle. This was not compliant with the OMG Data Distribution Service (DDS) standard API, version 1.4. Now the instance handles these APIs accept must correspond to a matching entity. In other words, the only valid inputs to these APIs are the handles returned by the aforementioned **DDS_DataWriter_get_matched_subscriptions** and **DDS_DataReader_get_matched_publications** APIs.

[RTI Issue ID CORE-9366]

5.12.4 Type used by <group_data>, <user_data>, and <topic_data> in XSD schema not compliant with DDS-XML specification

Release 6.1.0 updated the type used by <group_data>, <user_data>, and <topic_data>. The new representation is an xs:hexBinary value. The old format, in which the content is provided as a set of comma-separated decimal or hexadecimal elements values, will still be accepted by the Connext XML parser, except when the value contains a single byte in non-hexadecimal (0x) notation.

If a value is a single byte, the XML parser cannot tell whether the value uses the old format or the new one xs:hexBinary. To distinguish between them, a new Processing Instruction, <?rti-baseformat HEX|DEC?>, has been added. This Processing Instruction indicates whether the format is hexadecimal or decimal, and it is only needed when the value is a single byte in non-hexadecimal notation.

As a result of this change, an XML file that was valid in 6.0.1 may not be valid in 6.1.0. You may get an error similar to the following:

```
DDS_XMLQos_onEndOctetSequenceElement:Parse error at line XXX: processing instruction
<?rti-baseformat HEX|DEC?> missing
```

In particular, Infrastructure Services use XSD validation to verify the correctness of the input configuration file, and they will fail to load an XML configuration using the old format.

Pre-6.1.0 XML files that use a single byte value for <group_data>, <user_data>, and <topic_data> will need to be updated by adding the new Processing Instruction.

For more details, see the *Migration Guide* on the RTI Community Portal (https://-community.rti.com/documentation).

[RTI Issue ID CORE-9636]

5.12.5 Schema files were not compliant with DDS-XML spec

The following changes have been made to the schema files (rti_dds_profiles.xsd and rti_dds_qos_profiles.xsd and their included files) to make them compliant with the DDS-XML specification (https://www.omg.org/spec/DDS-XML/1.0/PDF):

- Renamed <participant_qos> to <domain_participant_qos>.
- Renamed <filter> to <content_filter>.
- Renamed <parameter_list> in <filter> to <expression_parameters> in <content_filter>.
- Renamed <value> in <parameter_list> to <element> in <expression_parameters>.

The old tags are still accepted by the Connext XML parser and the XSD schema to maintain backward compatibility.

The type of a union discriminator was also updated from a string with the pattern `(([a-zA-Z_]|::)([a-zA-Z_0-9]|::)*)` to an enumeration. This change was made to make the schema compliant with the latest DDS-XML specification (https://www.omg.org/spec/DDS-XML/1.0/PDF).

For example, the following XML snippet was compliant with the previous schema; it was valid because "myType" was compliant with the pattern:

```
<union ...>
    <discriminator type="myType"/>
</union>
```

Now, only the values defined in the "discriminatorTypeKind" enumerator type are valid:

```
<xs:simpleType name="discriminatorTypeKind">
    <xs:restriction base="xs:string">
        <xs:enumeration value="boolean"/>
        <xs:enumeration value="byte"/>
        <xs:enumeration value="int8"/>
        <xs:enumeration value="uint8"/>
        <xs:enumeration value="char8"/>
        <xs:enumeration value="char16"/>
        <xs:enumeration value="int16"/>
        <xs:enumeration value="uint16"/>
        <xs:enumeration value="int32"/>
        <xs:enumeration value="uint32"/>
        <xs:enumeration value="int64"/>
        <xs:enumeration value="uint64"/>
        <!-- Some other type -->
        <xs:enumeration value="nonBasic"/>
        ...
        </xs:enumeration>
    </xs:restriction>
</xs:simpleType>
```

See the schema files in **<installdir>/resource/schema**.

[RTI Issue ID CORE-9734]

5.12.6 Wrong GUID serialization for PID_DIRECTED_WRITE inline QoS parameter

The OMG RTPS 2.3 specification defines the GUID in a PID_DIRECTED_WRITE inline QoS parameter as a GUID_t. Previous releases incorrectly serialized the GUID as four integers in native endianness, as opposed to an array of 16 bytes. To conform with the specification, *Connext DDS* now serializes the GUID as an array of 16 bytes..

The specification also states that the PID_DIRECTED_WRITE inline QoS parameter should be ignored when interacting with protocol versions earlier than 2.4. Since *Connext DDS* announces an RTPS version of 2.3, other vendors should ignore this inline QoS parameter, and interoperability should not be affected.

To avoid breaking backward compatibility with previous releases, the GUID is still serialized natively when the remote *DomainParticipant* has an older *Connext DDS* product version.

[RTI Issue ID CORE-10122]

5.13 Fixes Related to Entities

5.13.1 Different value for reader_property_string_max_length/writer_property_string_max_length before and after creation of DomainParticipant

For the following QoS settings:

- **DDS_DomainParticipantResourceLimitsQosPolicy::reader_property_string_max_length**
- **DDS_DomainParticipantResourceLimitsQosPolicy::writer_property_string_max_length**

Connext DDS did not guarantee that the value of those settings before the creation of the *DomainParticipant* was the same as the value after calling **DDS_DomainParticipant_get_qos**.

For instance, the default value of those settings is 1024; however, after the creation of the *DomainParticipant*, if you called **DDS_DomainParticipant_get_qos**, the value returned was 1070.

This issue has been resolved. Now the value returned when calling **DDS_DomainParticipant_get_qos** is the same value that you specified in the creation of the *DomainParticipant*.

[RTI Issue ID CORE-10493]

5.13.2 Possible issues with communication and enabling DomainParticipant on Windows systems if network interface has multiple IP addresses

On Windows platforms, when a network interface has assigned more than one IP address, Connext DDS only detected the one with the lowest IP address. *DomainParticipants* running on a Windows host with this network configuration could not be discovered through the other IP addresses, causing communication issues.

Besides the communication issues, the *DomainParticipant* could not be enabled if the **allow/deny_interfaces_list** properties of the transport restricted the available IP addresses for Connext DDS to the ones not reported and there was no other transport enabled on that *DomainParticipant*.

This regression was introduced in 6.0.1 and is now fixed. Now, all the IP addresses of a network interface are detected and they will work as expected.

[RTI Issue ID CORE-11232]

5.14 Fixes Related to Vulnerabilities

This release fixes some potential vulnerabilities, including RTI Issue IDs CORE-10880 and CORE-10883.

5.15 Other Fixes

5.15.1 Some status fields not populated in Java callbacks

The following status fields were not populated by the Java language binding callbacks. Because of that, their values were incorrect:

- **SubscriptionMatchedStatus.current_count_peak**
- **PublicationMatchedStatus.current_count_peak**

This problem has been resolved.

[RTI Issue ID CORE-3095]

5.15.2 Unbounded memory growth when creating/deleting participants

The continuous creation/deletion of *DomainParticipants* could have led to unbounded memory growth, due to the fact that the deletion of a *DomainParticipant* left a small amount of memory behind.

This memory growth was not detected by memory profilers such as Valgrind because the leaked memory was reclaimed when deleting the DomainParticipantFactory before the application was closed.

This problem has been resolved.

[RTI Issue ID CORE-7881]

5.15.3 Possible unbounded memory growth when using Durable Writer History

In previous releases, the deletion of a *DataWriter* using durable writer history did not free all the memory associated with the *DataWriter*. This may have led to unbounded memory growth if you continuously created/deleted durable *DataWriters*.

Note that this memory growth was not reported as a memory leak by memory profilers such as Valgrind because all the memory was reclaimed when the process creating/deleting the *DataWriters* was shut down.

The problem has been resolved.

[RTI Issue ID CORE-8028]

5.15.4 Topic/Type regex typo in rtiddsspy summary display

There was a typo in the rtiddsspy summary window:

```
topic regex.....: *
topic regex.....: *
```

It is now corrected as follows:

```
topic regex.....: *
type regex.....: *
```

[RTI Issue ID CORE-8930]

5.15.5 Potential unbounded memory growth if DataWriter failed to publish data asynchronously

If a *DataWriter* failed to publish a sample asynchronously, the memory for that sample may not have been released until the *DataWriter* was deleted, leading to unbounded memory growth if the *DataWriter* failed to asynchronously publish an unbounded number of samples. Although it is unclear how the conditions for failure can occur in practice, the unbounded memory growth problem has been fixed. The sample can now be removed from the writer queue, and its memory can be released.

[RTI Issue ID CORE-9275]

5.15.6 RTI Admin Console showed wrong maximum annotation for uint64

Previously, if you used a uint64 (unsigned long long) in your type:

```
struct foo{
    uint64 u64;
};
```

RTI Admin Console would display the following IDL:

```
struct foo {
    @max(-1)
    uint64 u64; //@ID 0
};
//@Extensibility EXTENSIBLE_EXTENSIBILITY
```

The maximum annotation was not correct in this display. This issue affected only the IDL display in *Admin Console*.

This issue has been fixed, and now the IDL of this type will look as follows:

```
struct foo {
    uint64 u64; //@ID 0
};
//@Extensibility EXTENSIBLE_EXTENSIBILITY
```

[RTI Issue ID CORE-9846]

5.15.7 Potential application crash when receiving a sample on the service request channel

The reception of a sample over the service request channel (for instance, a TopicQuery request) may have caused an application to crash. The crash may have happened using non-C libraries if the related *DomainParticipant* had installed a listener that set the **onDataOnReaders** status on its mask. This issue has been fixed. Now the service request sample is processed properly.

This fix resolves a known issue in previous releases.

[RTI Issue ID CORE-9891]

5.15.8 Still reachable memory leaks: TransportMulticastMapping libraries were never unloaded

If you specified any mapping functions and their libraries in the TransportMulticast QosPolicy, those libraries were loaded but never unloaded. This problem has been fixed by unloading the libraries after they are used during *DomainParticipant* creation and *DataReader* creation.

> **Note:** You may still see "still reachable" memory leaks in "dlopen" and "dlclose". These leaks are a result of a bug in Valgrind (https://bugs.launchpad.net/ubuntu/+source/valgrind/+bug/1160352).

[RTI Issue ID CORE-9941]

5.15.9 Unused parameters in generated code

The functions **RTICdrStream_skipNonPrimitiveArray** and **RTICdrStream_skipNonPrimitiveSequence**, which appear in code generated by *Code Generator*, had an unused parameter. This parameter has been removed.

[RTI Issue ID CORE-10027]

5.15.10 Failure to allocate memory larger than 2 GB

Connext DDS failed to allocate heap memory larger than 2 GB. For example, if the DataWriterQos's **resource_limits.initial_samples** was large enough to cause a preallocation of more than 2 GB but less than the available heap memory, then *DataWriter* creation incorrectly failed. This problem has been fixed.

[RTI Issue ID CORE-10057]

5.15.11 [Java] checkPrimitiveRange failure using a type with float or double

In the Java API, when using a type with a float or a double, you may have seen the following error:

```
RTIXCdrInterpreter_checkPrimitiveRange:<type>:<field> deserialization error. Primitive value
<value> outside valid range [0,000000,
```

```
17976931348623157081452742373170435679807056752584499659891747680315726078002853876058955863276687817154045895351438246423432132688946418276846754670353751698604991057655128207624549009038932894407586850845513394230458323690322294816580855933212334827479782620414472316873817718091929988125040402618412485836 8,000000]
```

This failure occurred because the @min and @max default annotations for float and double in the Java API were not correct.

This problem has been resolved.

[RTI Issue ID CORE-10096]

5.15.12 Possible segmentation fault during DomainParticipant deletion

There was a rare race condition in which a segmentation fault occurred during *DomainParticipant* deletion. The segmentation fault occurred while the internal *Connext DDS* Event thread was calling the function **PRESInterParticipantWriter_write()**. This problem has been resolved.

[RTI Issue ID CORE-10099]

5.15.13 Typecodes with IDL representation greater than 1KB could not be printed using DDS_TypeCode_to_string APIs on Windows systems

TypeCodes whose IDL representation had a size greater than 1024 bytes could not be printed using the **DDS_TypeCode_to_string()** or **DDS_TypeCode_print_IDL()** APIs. This issue, which only affected Windows systems, has been resolved.

[RTI Issue ID CORE-10107]

5.15.14 XCDR2 serialization of a sample for a type with an optional primitive member may have been wrong in some cases

The XCDR2 serialization of a sample for a type with the following properties was incorrect:

- The type had a primitive member 'Pn' (it could be external but not optional) following another primitive member 'Pn-1' that was marked as optional
- The required alignment for 'Pn' was less than or equal to the required alignment for 'Pn-1'
- The optional member was set to NULL in the sample

This issue only affected the following language bindings: C, C++ (traditional and modern), and DynamicData in all languages.

It also affected all languages if using ContentFilteredTopics.

For example, the samples for the following types would not have been serialized correctly when the optional member was set to NULL:

```
struct MyType_1 {
    @optional int32 m1; /* alignment for int32 is 4 */
    int32 m2;  /* alignment for int32 is 4 */
};

struct MyType_1 {
    @optional int32 m1; /* alignment for int32 is 4 */
    double m2;  /* alignment for double is 4 */
};

struct MyType_1 {
    @optional int32 m1;  /* alignment for int32 is 4 */
    @external double m2;  /* alignment for double is 4 */
};
```

On x86 platforms, this issue only resulted in interoperability problems with other DDS vendors. For example, a sample serialized with *Connext DDS* would not have been deserialized correctly by other DDS implementations from different vendors.

On other platforms, such as Arm CPUs, this issue led to a bus error when deserializing.

This problem has been fixed.

[RTI Issue IDs CORE-10254 and CODEGENII-1302]

5.15.15 Memory leak using heap monitoring

When using heap monitoring, you may have had the following memory leak:

```
valgrind issue

==23698==  Address 0x779a345 is 21 bytes inside a block of size 31 alloc'd
==23698==    at 0x4C2FB55: calloc (in /usr/lib/valgrind/vgpreload_memcheck-amd64-linux.so)
==23698==    by 0x47D6FB: RTIOsapiHeap_reallocateMemoryInternal (heap.c:771)
==23698==    by 0x466847: REDAString_duplicate (String.c:1592)
==23698==    by 0x440136: REDAFastBufferPool_parseTypeName (fastBuffer.c:552)
==23698==    by 0x406215: REDAFastBufferTester_testParseTypeName (fastBufferTester.c:964)
```

This problem has been resolved.

[RTI Issue ID CORE-10284]

5.15.16 Several functions may have crashed when the "self" parameter was NULL (C API only)

Functions such as **DDS_DataReader_set_qos** may have crashed when the first argument ("self", the *DataReader*) was NULL.

This problem has been resolved and these functions now return DDS_RETCODE_BAD_PARAMETER in this situation.

[RTI Issue ID CORE-10353]

5.15.17 Samples not replaced when using Keep Last, Best Effort, finite max_samples, keyed data, and batching

Consider a scenario using BEST_EFFORT (in the RELIABILITY QosPolicy), KEEP_LAST (in the HISTORY QosPolicy), **max_samples**, keyed data, and batching. When using KEEP_LAST, a batch should never be dropped as long as it doesn't contain more samples than **max_samples/max_samples_per_remote_writer**. *Connext DDS* should replace samples that are currently in the queue with the samples in the batch.

However, when using BEST_EFFORT reliability, *Connext DDS* rejected a batch when **max_samples** was hit instead of making space in the queue, due to KEEP_LAST history replacement.

This scenario is now fixed. When a batch hits **max_samples**, *Connext DDS* makes space in the queue due to KEEP_LAST replacement.

[RTI Issue ID CORE-10580]

5.15.18 Memory leak when Skiplist function runs out of memory

If the internal function **REDASkiplist_newDefaultAllocator** ran out of memory, then entity creation would fail with a memory leak. Here is one example set of error messages, along with a valgrind result:

```
REDAFastBufferPool_growEmptyPoolEA: !allocate buffer of 48 bytes REDAFastBufferPool_
newWithParams:!create fast buffer pool buffers REDASkiplist_newDefaultAllocator:!create fast
buffer pool

==20092== 696 (256 direct, 440 indirect) bytes in 1 blocks are definitely lost in loss record 5
of 5
==20092==    at 0x4C2B975: calloc (vg_replace_malloc.c:711)
==20092==    by 0x1353974: RTIOsapiHeap_reallocateMemoryInternal (heap.c:793)
==20092==    by 0x131D8C8: REDASkiplist_newDefaultAllocator (SkiplistDefaultAllocator.c:287)
```

This problem has been fixed. Entity creation will now fail without a memory leak.

[RTI Issue ID CORE-10639]

5.15.19 Unexpected behavior in AsyncWaitSet operation if an invalid property was passed for its construction

Creating an AsyncWaitSet with an invalid property resulted in unexpected behavior. For example, calling **AsyncWaitSet::start()** could lead to a hang. This problem has been resolved.

[RTI Issue ID CORE-10657]

5.15.20 Segmentation fault when registering a type if error in operation

An application calling **DomainParticipant::register_type()** may have produced a segmentation fault if there was an error in this operation.

For example, this problem could have occurred when the type being registered was too big. In this case, you would have seen a set of error messages followed by a segmentation fault.

```
[0x0101D29C,0xBE497AA7,0x5553D2A0:0x000001C1{N=MyTypeParticipant,D=56}|REGISTER TYPE MyType]
RTIXCdrInterpreter_generateTypePluginProgram:failure generating serialize program for type
MyType: too many primitive values
[0x0101D29C,0xBE497AA7,0x5553D2A0:0x000001C1{N=MyTypeParticipant,D=56}|REGISTER TYPE MyType]
RTIXCdrInterpreterPrograms_generate:failure generating serialize program for type MyType
[0x0101D29C,0xBE497AA7,0x5553D2A0:0x000001C1{N=MyTypeParticipant,D=56}|REGISTER TYPE MyType]
RTIXCdrInterpreterPrograms_generateTopLevelPrograms:failure generating programs for type MyType
[0x0101D29C,0xBE497AA7,0x5553D2A0:0x000001C1{N=MyTypeParticipant,D=56}|REGISTER TYPE MyType]
RTIXCdrInterpreterPrograms_initializeWithParams:failure generating programs for type MyType
[0x0101D29C,0xBE497AA7,0x5553D2A0:0x000001C1{N=MyTypeParticipant,D=56}|REGISTER TYPE MyType]
DDS_TypeCodeFactory_assert_programs_w_parameters:ERROR: Failed to initialize resultPrograms
[0x0101D29C,0xBE497AA7,0x5553D2A0:0x000001C1{N=MyTypeParticipant,D=56}|REGISTER TYPE MyType]
DDS_TypeCodeFactory_assert_programs_in_global_list:!assert_programs
```

This issue has been fixed.

[RTI Issue ID CORE-10742]

5.15.21 Memory leak when Skiplist function ran out of memory, when using RTI Heap Monitoring

If the internal function **REDASkiplist_newDefaultAllocator** ran out of memory and you were using RTI Heap Monitoring, then entity creation failed with a memory leak. Here is an example set of error messages, along with a Valgrind result:

```
==12569== 57 bytes in 1 blocks are definitely lost in loss record 1 of 4
==12569==    at 0x4C2FB55: calloc (in /usr/lib/valgrind/vgpreload_memcheck-amd64-linux.so)
==12569==    by 0x499AB7: RTIOsapiHeap_reallocateMemoryInternal (heap.c:797)
==12569==    by 0x48158C: REDAString_duplicate (String.c:1563)
==12569==    by 0x4579D7: REDAFastBufferPool_parseTypeName (fastBuffer.c:535)
==12569==    by 0x457FBD: REDAFastBufferPool_newWithParams (fastBuffer.c:650)
==12569==    by 0x465639: REDASkiplist_newDefaultAllocator (SkiplistDefaultAllocator.c:301)
```

This problem has been fixed. Entity creation will now fail without a memory leak.

[RTI Issue ID CORE-10790]

5.15.22 Invalid serialization of samples with types containing nested structures with primitive members that require padding

In 6.0.1 and earlier, the serialization of samples with a type containing two or more levels of nested complex types, where the nested types have primitive members that require padding, may have failed. This means that a *DataReader* may have received an invalid value for a sample.

Example:

```
// Level-2 Nested type
struct Struct1 {
```

```
    uint8 m1;
    uint8 m2;
    int32 m3;
};
// Level-1 Nested type
struct Struct2 {
    int32 m1;
    int32 m2;
    uint8 m3;
    uint8 m4;
    Struct1 m5;
};
struct Struct3 {
    Struct2 m1;
};
```

In the above example, Struct2 and Struct1 are nested, and there is padding between Struct1::m2 (1-byte aligned) and Struct1::m3 (4-byte aligned) of 2 bytes.

This issue only applied to nested types that are appendable or final for XCDR1 data representation or final for XCDR2 data representation.

This problem affected DynamicData and the generated code for the following languages: C, C++, C++03, and C++11.

For generated code, a potential workaround to this problem was to generate code with a value of 1 or 0 for the **-optimization**, but this may have had performance implications.

This problem has been resolved.

[RTI Issue IDs CORE-10820 and CODEGENII-1486]

5.15.23 java.lang.ClassCastException on a DataReader subscribing to a Topic of the String builtin type

A *DataReader* subscribing to a *Topic* associated with the String builtin type may have thrown a `java.lang.ClassCastException` after receiving samples or creating a QueryCondition if:

1. The *DataReader* set the **dds.data_reader.history.memory_manager.fast_pool.pool_buffer_max_size** to a value greater than or equal to 0.
2. And, the *DataReader* was a ContentFilteredTopic or was creating QueryConditions.

This problem has been resolved.

[RTI Issue ID CORE-10796]

5.15.24 Segmentation fault when deserializing a large sequence of locator filters

If the sequence of locator filters had a length longer than the maximum value of a 32-bit signed integer (2147483647), then deserialization did not behave correctly. As a result, the internal function **PRESLocatorFilterQosProperty_copy()** would crash due to attempting to dereference a NULL pointer. This problem has been resolved.

[RTI Issue ID CORE-10958]

5.15.25 Segmentation fault when large numbers of resources were being allocated

There was an issue in an underlying data structure in the *Connext DDS* libraries which could cause corruption and lead to a crash when allocating large pieces of memory. This could happen, for example, if your application was writing or storing thousands of samples or instances. This issue has been resolved.

[RTI Issue ID CORE-11035]

5.15.26 Crash printing a typeCode in Java

There was a crash in Java when printing a data type, which had been propagated through discovery using TypeCode, with the following exception:

```
ava.lang.NullPointerException: Cannot invoke
"com.rti.dds.typecode.AnnotationParameterValue.discriminator()" because the return value of
"com.rti.dds.typecode.Annotations.default_annotation()" is null
        at com.rti.dds.typecode.TypeCode.print_default_literal_annotations(TypeCode.java:1848)
        at com.rti.dds.typecode.TypeCode.print_IDL(TypeCode.java:2675)
        at com.rti.dds.typecode.TypeCode.print_complete_IDL(TypeCode.java:2500)
        at msgSubscriber$BuiltinPublicationListener.on_data_available(Unknown Source)
        at com.rti.dds.subscription.DataReaderListenerImpl.on_data_available
(DataReaderListenerImpl.java:158)
```

The exception occurred when the type contained an enum.

This issue has been fixed, and now the TypeCode can be printed.

[RTI Issue ID CORE-11071]

5.15.27 min and max annotations (@min and @max) incorrectly displayed for float and double types when IDL was viewed in Admin Console

The default minimum and maximum annotations (@min and @max) are no longer printed when an IDL representation of a type is viewed in *RTI Admin Console*.

[RTI Issue ID CORE-11101]

5.15.28 Possible segmentation fault on some architectures in release mode while writing an unbounded string type

When using the release libraries for some architectures, writing a sample of a type that contains an unbounded string may have led to a segmentation fault in the internal function **RTIXCdrInterpreter_getSerSampleSize()**. This problem, which only affected Connext DDS 6.0.0 and above, has been fixed.

[RTI Issue ID CORE-11141]

5.15.29 Application may not have received samples published by more than two DataWriters working as Collaborative DataWriters

An application may not have received samples published by multiple *DataWriters* working as Collaborative DataWriters.

This issue occurred only when the following two conditions were met:

- The *DataReader* configured **availability.max_data_availability_waiting_time** or **availability.max_endpoint_availability_waiting_time** with a value greater than zero.
- The *DataWriters* did not send virtual heartbeats

For example, assume that two *DataWriters* were configured with the same virtual GUID, and they published two samples with sequence numbers 1 and 3. Sequence number 2 is missing. In this example, a *DataReader* would have received the sample with sequence number 1 but would have never received the sample with sequence number 3.

This issue could also affect infrastructure services that run in collaborative mode, such as *RTI Routing Service* and *RTI Persistence Service*. For example, the two *DataWriters* in this example could correspond to the *DataWriters* of two *Routing Services* that run in redundant mode. In the case of *Persistence Service*, the two *DataWriters* could have been the original *DataWriter* and the *Persistence Service DataWriter*.

This issue has been resolved.

[RTI Issue ID CORE-11200]

5.15.30 Potential crash upon receiving a corrupted RTPS CRC32 submessage

In previous releases, receiving a corrupted RTPS CRC32 submessage (for example, as a consequence of network corruption) may have triggered a crash. This problem has been resolved, and receiving a corrupted RTPS CRC32 submessage should no longer trigger a crash.

[RTI Issue ID CORE-11286]

5.15.31 Segmentation fault while deleting participant when monitoring enabled in separate domain ID

The deletion of a *DomainParticipant* in which monitoring was enabled in a separate domain ID (through the property **rti.monitor.config.new_participant_domain_id**) may have led to a segmentation fault.

This problem has been resolved.

[RTI Issue ID MONITOR-291]

5.15.32 Requester may have received spurious replies when replier was deleted

A requester waiting for a reply to a particular request may have been unblocked due to the reception of a not-alive-no-writers message after a replier was deleted or lost liveliness. This unexpected behavior forced application code to wait in a loop to ignore these messages.

This problem has been resolved in all Request-Reply APIs (C, Traditional C++, Modern C++, Java, and .NET). The not-alive-no-writers message is now purged in *Requesters* and *Repliers*, and will no longer be received or unblock a *Requester* waiting for replies.

[RTI Issue ID REQREPLY-63]

5.15.33 Interoperability issue between Java/NET/DynamicData and C/C++/modern C++ applications when using keyed types and XCDRv1 encapsulation

In 6.0.0 and 6.0.1, the instance keyhash for a keyed type using XCDR (Extensible CDR version 1) encapsulation was calculated differently in the Java, .NET, and DynamicData languages when the code for the keyed type was generated using the **-optimization 0** option and the keyed type contained one key member whose type was a typedef of a struct/valuetype type in which only some of the members were marked as **@key** fields.

For example:

```
struct SimpleKeyedType
{
    @key octet m1;
    octet m2;
};
typedef SimpleKeyedType SimpleKeyedTypeAlias;
struct KeyedType
{
    @key SimpleKeyedTypeAlias m1;
};
```

The right calculation was done in Java.

As a result, the subscribing application might have observed some unexpected behavior related to instances. Specifically, the call to **DataReader::lookup_instance()** might have failed and returned HANDLE NIL even if the instance was received.

This also affected compatibility with the languages C, C++, and Modern C++ in 5.3.1 or earlier releases.

This problem has been resolved.

[RTI Issue IDs CORE-11290 and CODEGENII-1485]

5.15.34 Samples lost on DataReader because max_instances was exceeded were not AppAcked

This issue was resolved in 6.0.1, but not documented at that time.

In previous releases where application-level acknowledgment was used, samples dropped from the *DataReader* because **max_instances** was exceeded were not automatically AppAcked. This may have lead to scenarios in which samples were never removed from a *DataWriter's* sample queue.

[RTI Issue ID CORE-11082]

Chapter 6 Known Issues

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 DDS 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 DDS* 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 DDS 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 DDS* 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 DDS 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 DDS*. 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.

[RTI Issue ID CORE-3018]

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 **AvailabilityQosPolicy.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 DDS* 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 DDS 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 DDS Core Libraries User's Manual*.

[RTI Issue ID CORE-5360]

6.7.3 Discovery with Connext DDS Micro fails when shared memory transport enabled

Given a *Connext DDS* application with the shared memory transport enabled, a *Connext DDS* Micro 2.4.x application will fail to discover it. This is due to a bug in *Connext DDS* Micro that prevents a received participant discovery message from being correctly processed. This bug will be fixed in a future release of *Connext DDS* Micro. As a workaround, you can disable the shared memory transport in the *Connext DDS* 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 DDS* 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 DDS Micro*, so this approach will not work if *Connext DDS 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 DDS 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 DDS 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 DDS* 6.1.0 receives coherent sets from *Connext DDS* 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 DDS* 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 DDS Micro 3.0.3 installation package currently compatible only with Connext 6.0.1 installer

Connext DDSMicro 3.0.3 must be installed with *Connext DDSProfessional* release 6.0.1. It cannot be installed with release 6.1.0. *Connext DDSMicro* 3.0.3 can communicate with either release. Customers

licensing *Connext DDSMicro* will be notified when a *Connext DDSMicro* release that is compatible with the 6.1.0 installer is available.

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://bugs.launchpad.net/ubuntu/+source/valgrind/+bug/1160352).

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

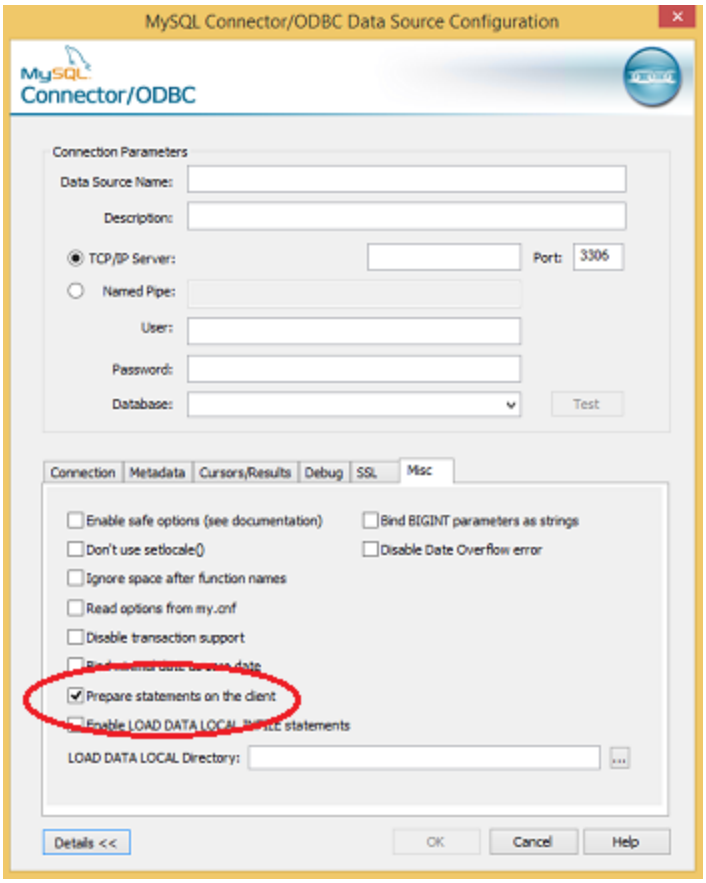
[RTI Issue IDs CORE-9941, SEC-1026, and COREPLG-510]

6.13.2 'Incorrect arguments to mysqld_stmt_execute' errors when using MySQL ODBC driver

Some versions of the MySQL ODBC driver may not work out-of-the-box and produce ODBC errors that include the following message:

```
Incorrect arguments to mysqld_stmt_execute.
```

In this case, you will need to enable the "Prepare statements on the client" option in the DSN configuration. You will find that option under **Details, Misc, Prepare statements on the client** when adding or configuring a DSN. This behavior has been observed with MySQL ODBC driver version 8.0.23, but other versions may also be affected.



6.13.3 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]

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/.