RTI Connext DDS

Core Libraries

What's New in

Version 5.2.3

Trademarks

Copy and Use Restrictions

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/

What's New in 5.2.3

1 New Platforms 1

2 Ability to Install Packages and Run Services on Windows Platform without "My Documents" 2

3 Support for Unbounded Built-in Types in Java API 2

4 Support in Java and C# DynamicData APIs to Convert to/from CDR Buffer 3

5 Ability to Log Messages in Distributed Logger with Specific Timestamps 3

6 Distributed Logger Option to Prevent Forwarding of Infrastructure-Related Messages 3

7 New PropertyQosHelper APIs 4

8 Some Infrastructure Types now Include Overloaded Operator to Print to Output Stream—Modern C++ API Only 4

9 Support for Statically Linking RTI TLS Support Libraries with TCP Transport Plugin 4

10 Microsoft Redistributable Libraries now Packaged with Connext DDS Libraries for Windows Targets 5

11 Ability to Define Data Types in XML Schemas (XSD) 5

12 Functions from_cdr_buffer() and to_cdr_buffer() Improved and Moved—Modern C++ API Only 6

13 New DataWriter and DataReader Properties for Backward Compatibility with 4.2e 6

What's New in 5.2.3

Connext DDS 5.2.3 is a maintenance release based on feature release 5.2.0. This document high-lights new platforms and improvements in 5.2.3.

For what's fixed in 5.2.3, see the RTI Connext DDS Core Libraries Release Notes.

For what's new and fixed in 5.2.0, see the *What's New* and *Release Notes* documents provided with 5.2.0, respectively.

1 New Platforms

This release adds support for the following platforms:

Table 1.1 New Platforms in 5.2.3

Operating System	CPU	Compiler	RTI Architecture Abbreviation
Android 5.0, 5.1	ARMv7A	gcc 4.9 (NDK r10e)	armv7aAndroid5.0gcc4.9ndkr10e
		Java Platform, Standard Edition JDK 1.7 or 1.8	
CentOS 7.0	x86	gcc 4.8.2	i86Linux3gcc4.8.2
		Java Platform, Standard Edition JDK 1.7 or 1.8	
	x64	gcc 4.8.2	x64Linux3gcc4.8.2
		Java Platform, Standard Edition JDK 1.7 or 1.8	

Operating System	CPU	Compiler	RTI Architecture Abbreviation
Debian Linux 3.12 (Available only with custom support)	ARMv7a Cortex-a9	gcc 4.9.3	armv7aLinux3.12gcc4.9.3cortex-a9
iOS 8.2	Dual-Core 64-bit Apple A7	clang 6.1	arm64iOS8clang6.1
	x86	clang 6.1	x86_64iOS8clang6.1
Red Hat Enterprise Linux 6.7	x86	gcc 4.4.5	i86Linux3gcc4.4.5
		Java Platform, Standard Edition JDK 1.7 or 1.8	
	x64	gcc 4.4.5	x64Linux3gcc4.4.5
		Java Platform, Standard Edition JDK 1.7 or 1.8	
OS X 10.11	x64	clang 7.0	x64Darwin15clang7.0
		Java Platform, Standard Edition JDK 1.7 or 1.8	
Windows 10	x86	VS 2015	i86Win32VS2015
	x64	VS 2015	x64Win64VS2015

2 Ability to Install Packages and Run Services on Windows Platform without "My Documents"

This release introduces the ability to install packages and run services from an account without a "My Documents" directory on a Windows platform.

3 Support for Unbounded Built-in Types in Java API

This release adds support for unbounded built-in types in the Java API (it already existed for C, C++, and .NET).

When unbounded support is configured, the middleware will not preallocate the *DataReader* queue's samples to their maximum size. Instead, it will deserialize incoming samples by dynamically allocating and deallocating memory to accommodate the actual size of the sample value.

See Section 3.2.7.2, Unbounded Built-in Types, in the *RTI Connext DDS Core Libraries User's Manual*.

4 Support in Java and C# DynamicData APIs to Convert to/from CDR Buffer

Support for the DynamicData's **to_cdr_buffer()** and **from_cdr_buffer()** APIs has been extended to the Java and C# languages. See the API Reference HTML documentation for more information.

5 Ability to Log Messages in Distributed Logger with Specific Timestamps

The message timestamp was automatically calculated when a message was logged through Distributed Logger. To allow logging messages with arbitrary timestamps, there is a new log method that receives an object encapsulating all message parameters:

C method name:

```
void RTI_DL_DistLogger_logMessageWithParams(RTI_DL_DistLogger *self,const struct RTI_DL_
DistLogger_MessageParams *params)}}
```

C++ method name:

```
void RTI_DLDistLogger::logMessageWithParams(const struct RTI_DL_DistLogger_MessageParams &
params)
```

Java method name:

```
public final class DistLogger {
  public void log(LogMessageData messageData);
}
```

For more details, see the API Reference HTML documentation.

6 Distributed Logger Option to Prevent Forwarding of Infrastructure-Related Messages

In previous releases, if an RTI Logger device was set prior to Distributed Logger singleton initialization, the Logger device was silently overwritten by the singleton. This caused Distributed Logger to publish all log messages generated by infrastructure code (i.e. both the middleware and tools).

There is a new Distributed Logger option, **logInfrastructureMessages** (or **<log_intrastructure_messages>** in XML) that allows you to modify this behavior:

- If set to true (the default), Distributed Logger's singleton will overwrite the existing RTI Logger device and Distributed Logger will forward infrastructure-generated messages.
- If set to false, Distributed Logger will not forward any infrastructure messages and will leave the RTI Logger device untouched.

This new option is available in the C, C++ and Java APIs. For more information, see the API Reference HTML documentation.

7 New PropertyQosHelper APIs

There are new C/C++ APIs in PropertyQosHelper. These APIs allow you to set properties with a pointer as their value. Additionally, new utility APIs have been added to the modern C++ API. The added APIs are:

New C APIs:

```
DDS_ReturnCode_t DDS_PropertyQosPolicyHelper_assert_pointer_property(
        struct DDS_PropertyQosPolicy *policy,
        const char *name,
        const void *pointer);

DDS_ReturnCode_t DDS_PropertyQosPolicyHelper_add_pointer_property(
        struct DDS_PropertyQosPolicy *policy,
        const char *name,
        const void *pointer);
```

New C++ APIs:

```
DDS_ReturnCode_t DDSPropertyQosPolicyHelper::assert_pointer_property(
        DDS_PropertyQosPolicy& policy,
        const char *name,
        const void *pointer);
DDS_ReturnCode_t DDSPropertyQosPolicyHelper::add_pointer_property(
        DDS_PropertyQosPolicy& policy,
        const char *name,
        const void *pointer);
```

New Modern C++ APIs:

```
std::string ptr_to_str(const void * p);
void * str_to_ptr(const std::string& s);
```

8 Some Infrastructure Types now Include Overloaded Operator to Print to Output Stream—Modern C++ API Only

The following types now define an overloaded **operator<<** to print to an **std::ostream**: **SampleIdentity**, **SequenceNumber**, **Guid** and **Cookie** (all in the **rti::core namespace**).

9 Support for Statically Linking RTI TLS Support Libraries with TCP Transport Plugin

This release adds the ability to statically link the RTI TLS Support Libraries with the TCP Transport plugin.

In order to support this, two new properties have been added to the TCP Transport Plugin: **tls_create_function_ptr** and **tls_delete_function_ptr**. The following code snippet shows how to change a *DomainParticipant's* QoS to use the TCP Transport plugin and RTI TLS Support with the static libraries:

```
/* Set TCP creation function from the static library */
retcode = DDSPropertyQosPolicyHelper::assert_pointer_property(
    participant_qos.property,
    "dds.transport.TCPv4.tcp1.create_function_ptr",
    (void*)NDDS_Transport_TCPv4_create);

if (retcode != DDS_RETCODE_OK) {
    printf("set tcp create function ptr failed %d\n", retcode);
    return -1;
}

/* Set TLS Support creation function from the static library */
retcode = DDSPropertyQosPolicyHelper::assert_pointer_property(
    participant_qos.property,
    "dds.transport.TCPv4.tcp1.tls_create_function_ptr",
    (void*)RTITLS_ConnectionEndpointFactoryTLSv4_create);


if (retcode != DDS_RETCODE_OK) {
    printf("set tls create function ptr failed %d\n", retcode);
return -1;
}

/* Set TLS Support delete function from the static library */
retcode = DDSPropertyQosPolicyHelper::assert_pointer_property(
    participant_qos.property,
    "dds.transport.TCPv4.tcp1.tls_delete_function_ptr",
    (void*)RTITLS_ConnectionEndpointFactoryTLSv4_delete);

if (retcode != DDS_RETCODE_OK) {
    printf("set tls delete function ptr failed %d\n", retcode);
return -1;
}
```

10 Microsoft Redistributable Libraries now Packaged with Connext DDS Libraries for Windows Targets

In some cases, a Java developer might have a dependency on the Connext DDS target libraries and not have the correct version of the Microsoft Visual Studio Redistributable libraries on their development or target machine. These libraries are now shipped with the appropriate libraries for Windows target systems.

11 Ability to Define Data Types in XML Schemas (XSD)

This release introduces support for input files in XSD format to the Code Generator. Section 3.5 in the *User's Manual* explains how to describe data types with XML schemas (XSD).

There is also a new option for the Code Generator (*rtiddsgen*) to transform IDL files into XSD files: **-convertToXsd**. See the *Code Generator User's Manual* for more information.

12 Functions from_cdr_buffer() and to_cdr_buffer() Improved and Moved–Modern C++ API Only

In previous releases, the functions that allow you to serialize/deserialize a sample of an IDL type *T* to and from a CDR byte buffer were defined in the structure **topic_type_support<*T*>**. For DynamicData, those functions were in the namespace **rti::core::xtypes**. For the built-in types (StringTopicType, etc.) those functions were missing (CORE-6843).

Now these functions are generic standalone functions in the **rti::topic** namespace, in a new header file, **rti/topic/cdr.hpp**. In addition, two versions of **from_cdr_buffer()** are provided:

- **template <typename TopicType> void from_cdr_buffer_no_alloc(TopicType& sample, const std::vector<char>& buffer)**
- **template <typename TopicType> TopicType from_cdr_buffer(const std::vector<char>& buffer)**

The second one is new and returns a sample. The first one reuses an existing sample.

13 New DataWriter and DataReader Properties for Backward Compatibility with 4.2e

To interoperate with 4.2e or lower, previously you only had to use the **-use42eAlignment** flag when generating code with *rtiddsgen*. Starting with Connext DDS 5.2.3, you *also* need to set the following *DataWriter* and *DataReader* properties to "1":

- *DataWriter*: **dds.data_writer.type_support.use_42e_alignment**
- *DataReader*: **dds.data_reader.type_support.use_42e_alignment**

See also: double, long long, unsigned long long or long double Wire Compatibility.