RTI Connext

Core Libraries

What's New in Version 7.0.0

Notices

Deprecations and Removals

Any deprecations or removals noted in this document serve as notice under the Real-Time Innovations, Inc. Maintenance Policy #4220 and/or any other agreements by and between RTI and customer regarding maintenance and support of RTI’s software.

Deprecated means that the item is still supported in the release, but will be removed in a future release. *Removed* means that the item is discontinued or no longer supported. By specifying that an item is deprecated in a release, RTI hereby provides customer notice that RTI reserves the right after one year from the date of such release and, with or without further notice, to immediately terminate maintenance (including without limitation, providing updates and upgrades) for the item, and no longer support the item, in a future release.

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

Introduction

RTI® Connext® 7.0.0 is an early access release. This document highlights new features, platforms, and improvements in the Core Libraries for 7.0.0.

For what's *fixed* in the Core Libraries for 7.0.0, see the RTI Connext Core Libraries Release Notes.

For backward compatibility information between 7.0.0 and previous releases, see the *Migration Guide* on the RTI Community Portal (https://community.rti.com/documentation).

Table 1.1 Release 7.0.0 Highlights (these and more features are described below)

	Ability to partition groups of applications Now create groups of applications at the *DomainParticipant* level to reduce network traffic, CPU, and memory utilization.
	DDS Spy and DDS Ping Improvements Debug applications from the command line faster with improved *DDS Spy* output. Use filters to display discovery data, user data, or everything.
	DDS-XML format for Limited Bandwidth Endpoint Discovery Plugin Any DDS-XML description you already have of your system can now be used directly for the LBED plugin, without needing to translate it to another schema.
	Faster Debugging with New Logging Categories Two new logging categories, for discovery and security activity, enable you to more easily filter and trace discovery operations and *RTI Security Plugins* operations. See Security Improvements and Logging Improvements.
	Reduce network bandwidth utilization by allowing Connext to limit allowed interfaces even further A new **max_interface_count** field limits the number of network interfaces used by a *DomainParticipant*.

	Reorganized Core Libraries User's Manual for easier browsing A reorganized *RTI Connext Core Libraries User's Manual* is easier to use for people who are new to the Connext ecosystem.
	Enhanced scalability for peer-to-peer communications with new Simple Participant Discovery Protocol 2.0 (experimental) For systems with a large number of DomainParticipants, the SPDP 2.0 option improves scalability by sending optimized matching and liveliness messages among *DomainParticipants*.
	Connext Python API adds support for user data types and code generation (experimental) Get started with Python faster for your application's data types. Now use *Code Generator* to generate code in Python.

For what's new and fixed in other products included in the *Connext* suite, see those products' release notes on https://community.rti.com/documentation or in your installation. Or find those products' release notes here:

- RTI Code Generator Release Notes
- RTI Routing Service User's Manual
- RTI Recording Service User's Manual
- RTI Persistence Service Release Notes
- RTI Web Integration Service Release Notes
- RTI Launcher Release Notes
- RTI Admin Console Release Notes
- RTI Monitor Release Notes
- RTI Shapes Demo User's Manual
- RTI System Designer Getting Started Guide
- RTI Real-Time WAN Transport Release Notes
- RTI Security Plugins Release Notes
- RTI Limited Bandwidth Plugins Release Notes
- RTI Limited Bandwidth Endpoint Discovery Plugin Release Notes
- RTI Cloud Discovery Release Notes

See also 13 Product Availability Changes on page 30.

1 Discovery Scalability and Troubleshooting

1.1 Enhanced scalability for peer-to-peer communications with new Simple Participant Discovery Protocol 2.0 (experimental)

The new Simple Participant Discovery Protocol (SPDP) 2.0 improves scalability by reducing the amount of redundant information that is sent during participant discovery and to maintain participant liveliness. SPDP 2.0, which is experimental in this release, accomplishes this improvement by splitting participant discovery into separate phases. First, only the information that is required to make a match is sent periodically in a bootstrap phase (e.g., domain ID, partition). Second, if the two participants match based on their bootstrap information, the complete information is sent (e.g., properties, participant name) over a reliable channel so that it is not repeated unless there is a change. Finally, after two participants have exchanged bootstrap and configuration data, a periodic lightweight message is used to maintain liveliness.

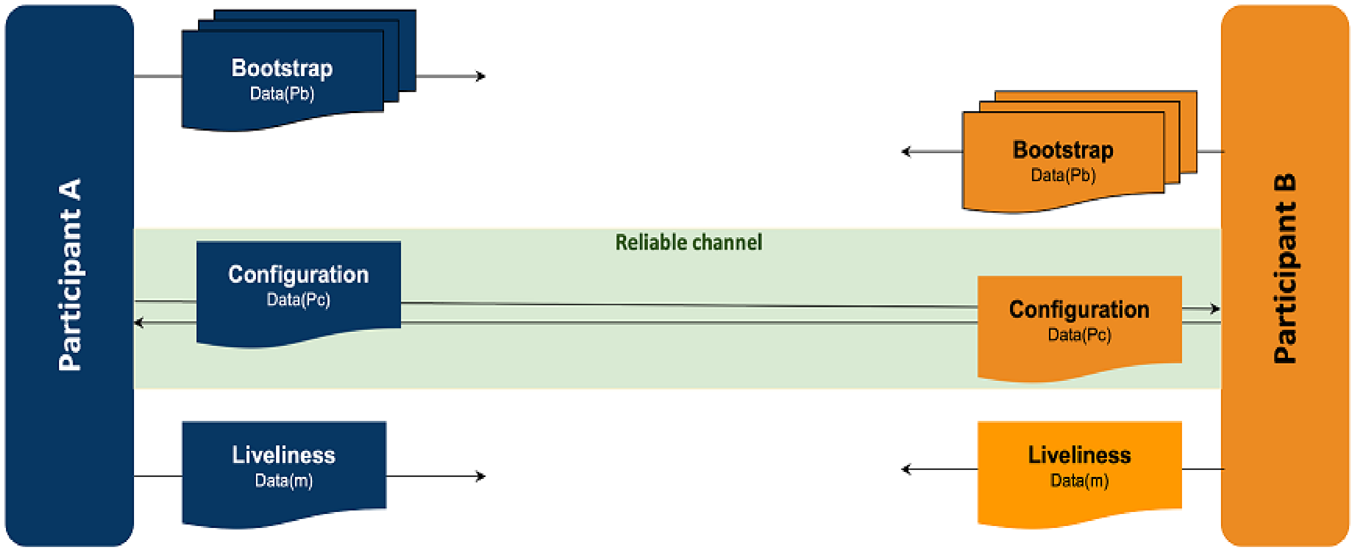
SPDP 2.0, which is experimental in this release, is not enabled by default. To enable it, set the **builtin_discovery_plugins** field in the DISCOVERY_CONFIG QosPolicy to (see the *DISCOVERY_CONFIG QosPolicy* section of the RTI Connext Core Libraries User's Manual) SPDP2 | SEDP. This will enable SPDP 2.0 and the existing Simple Endpoint Discovery Protocol.

In the current SPDP, participants send out participant announcements (also known as participant DATA submessages) that contain all of the information that a participant needs to communicate with remote participants. This includes information that is necessary in order to establish communication, like the domain ID and locators at which this participant can be reached, information that is needed only if the participants will continue with endpoint discovery, and information that is not strictly needed at all and is only useful for debuggability/observability, like system information properties. All of this information can add up to messages that are around 2k or more, and they are sent not just to initially discover participants but to maintain liveliness with them. The size can be a problem, particularly in systems that are trying to avoid IP fragmentation by setting the DDS maximum transmission unit (MTU) to a small value (<1500) and using DDS-layer fragmentation.

SPDP 2.0 solves this problem by splitting the participant DATA submessages into three:

- Bootstrap (DATA(Pb)) messages, which send only the information required to make a match.
- Configuration (DATA(Pc)) messages, which send complete information only once a match is confirmed.
- Lightweight liveliness (DATA(m)) messages to maintain liveliness with a participant.

Figure 1.1: Simple Participant Discovery Protocol 2.0 (Experimental)



The experimental version of the protocol also does not stop sending periodic bootstrap messages to a participant once discovery has been established with that participant. In upcoming feature releases, ongoing bandwidth consumption will be further reduced by stopping the periodic announcement of bootstrap messages to locators at which there is a discovered participant. Discovered participants will only receive lightweight, periodic liveliness messages, sent at the **DiscoveryConfigQosPolicy.participant_liveliness_assert_period**, and a single, reliable message whenever there is an update to a participant's configuration, such as a partition or locator change.

See the *(Experimental) Simple Participant Discovery 2.0* section of the RTI Connext Core Libraries User's Manual for further details.

1.2 Improved bandwidth usage through improvements to existing Simple Participant Discovery Protocol

There have been a number of improvements made to the existing Simple Participant Discovery Protocol to reduce bandwidth usage as new participants are discovered and updated.

Previously, when a new remote participant was discovered by a local participant, the local participant would respond back to the new remote participant as well as all previously discovered participants with **DiscoveryConfigQosPolicy.initial_participant_announcements** number of announcements. Most of this was unnecessary traffic, since the previously discovered participants already had the information.

Now, when a new remote participant is discovered, the local participant sends its response participant announcements directly back to the new participant and not to any previously discovered participants. This improvement will be useful in systems that are using unicast discovery instead of multicast. In systems that are using multicast, the same number of multicast packets are sent whether there were already previously discovered participants or not.

The other new feature is that the number of packets that are sent to a new remote participant is configured using a new **DiscoveryConfigQosPolicy.new_remote_participant_announcements** QoS setting instead of **initial_participant_announcements**. The default for **new_remote_participant_announcements** is 2, while the default value for **initial_participant_announcements** is 5. The benefit of having separate configuration values is that a participant can send out a larger number of announcements over a longer duration of time during startup to ensure better success at discovering all other applications successfully, configured using **initial_participant_announcements**, and then send fewer announcements after the initial startup period in response to new applications that join the system.

The final improvement is that a participant will not send its participant announcement to a remote participant after the remote participant has announced a locator change. Because the local participant's configuration has not changed, there is no need to send a response to a change in a remote participant.

1.3 View status of all discovered entities in your system using discovery snapshots

You can now take a discovery snapshot representing the discovery status of your system, using the **take_snapshot** APIs. The discovery snapshot is useful for debugging discovery issues. You can get information about the discovery status among *DomainParticipants*, in addition to compatible and incompatible matches for *DataWriters* and *DataReaders*.

The output information can be printed to a file (if you provide a file name to the API) or through the *Connext* builtin logging system (if you do not provide a file name to the API).

An example output, for *DomainParticipants,* is as follows:

```
-------------------------------------------------------------------------
-------------------------------------------------------------------------
Participant guid="0x0101F2F1,0xC229B376,0x46572559:0x00000000"
domain_id=0 name="participantTestName" role="participantTestRole"
-----------------------------
Matched Participants:
-----------------------------
guid="0x0101D75E,0xB70D1850,0x2B0D229B:0x00000000"
name="participantTestName" role="participantTestRole"
unicastLocators="udpv4://10.70.2.68:7413
shmem://CA1B:28DA:1E18:F955:3727:3AFE:0000:0000:7413"
-------------------------------------------------------------------------
-------------------------------------------------------------------------
```

See the *Discovery Snapshots* section of the RTI Connext Core Libraries User's Manual for more details.

1.4 Troubleshoot discovery issues faster using new DISCOVERY logging category

A new logging category, NDDS_CONFIG_LOG_CATEGORY_DISCOVERY, has been introduced in this release. It enables you to filter and more easily trace discovery-related operations performed by

Connext. Some discovery-related log messages have also been improved to provide more detail.

For information on filtering log messages by category, see the *Configuring Connext Logging* section of the RTI Connext Core Libraries User's Manual.

2 Language Bindings, APIs, XML Configuration

2.1 Connext Python API adds support for user data types and code generation (experimental)

This release includes a major update for the experimental *Connext* Python API. (Note: this API shouldn't be confused with *Connector for Python*; this is a full *Connext* API.)

This release adds support for publishing and subscribing to user data types. These data types can be generated from IDL, XML, and XSD, or they can be defined in the user application as decorated dataclasses.

The following is a full and working application that publishes the type InventoryItem:

```
// MyInventoryPublisher.py
import rti.connextdds as dds
import rti.idl as idl

@idl.struct
class InventoryItem:
    name: str = ""
    price: float = 0.0
    quantity: int = 0


participant = dds.DomainParticipant(domain_id=0)
topic = dds.Topic(participant, "MyInventory", InventoryItem)
publisher = dds.Publisher(participant)
writer = dds.DataWriter(publisher, topic)

writer.write(InventoryItem(name="Apple", price=0.45, quantity=10))
writer.write(InventoryItem(name="Orange", price=0.35, quantity=8))
writer.write(InventoryItem(name="Banana", price=0.25, quantity=6))
```

The type InventoryItem can be defined in Python as above, or it can be defined in IDL as follows:

```
// MyInventory.idl
struct InventoryItem {
    string name;
    double price;
    int64 quantity;
};
```

And then *rtiddsgen* can generate the Python type with the following command:

```
rtiddsgen -language Python -example universal MyInventory.idl
```

The *Connext* Python API is still considered experimental in this release. A production-ready release is expected in the near future.

2.2 Use Request-Reply communication pattern with the new C# API

When the new C# API was first released in 6.1.0, it did not include support for the Request-Reply communication pattern. Request-Reply functionality is now fully supported by the C# API. The new Request-Reply API has also been redesigned to be used more intuitively and to follow modern C# best-practices.

For more information, see the C# API Reference on the RTI Community Portal (https://-community.rti.com/documentation) and this code example.

2.3 Free DDS thread-specific storage on demand in new C# API

The first release of the new C# API did not provide a method to free DDS thread-specific storage on demand. This is now possible via the new ThreadManager disposable object.

2.4 All events in DomainParticipants, Publishers, and Subscribers now available for use in new C# API

The first release of the new C# API did not include support for events (listeners) in the *DomainParticipant* and *Publisher*, and only supported DataOnReaders events in the *Subscriber*. All events in *DomainParticipant*, *Publisher,* and *Subscriber* are now available for use.

2.5 Redirect logging output to a custom handler in Modern C++ API

In previous releases, the Modern C++ API didn't provide a way to redirect the *Connext* logging output to an "output device" for custom processing. This release adds **rti::config::Logger::output_handler()** and **rti::config::Logger::reset_output_handler()**.

For example:

```
std::vector<std::string> saved_logs;
Logger::instance().output_handler([&saved_logs](const LogMessage& message) {
    saved_logs.push_back(message.text);
});
```

2.6 Get list of remote DomainParticipants with a given participant name

It is sometimes desirable to identify remote *DomainParticipants* by their participant name (see the *ENTITY_NAME QosPolicy* section of the RTI Connext Core Libraries User's Manual). But many of the current *DomainParticipant* API methods, such as **DomainParticipant::ignore_participant()**, identify *DomainParticipants* by their InstanceHandle_t.

In this release, we bridge the gap between InstanceHandle_t and participant names:

- If you know the participant name of the *DomainParticipant* that you want to ignore, and you need to get the associated InstanceHandle_t, then you can use a new API method called **DomainParticipant::get_discovered_participants_from_subject_name()**. Pass the API a participant name string, and it outputs an InstanceHandleSeq of *DomainParticipants* that have this string as their value for **ParticipantBuiltinTopicData::participant_name.name**.
- In addition, if you know the InstanceHandle_t of a *DomainParticipant* for which you want to get the participant name, you can use another new API method called **DomainParticipant::get_discovered_participant_subject_name()**.

For more information, see the API Reference HTML documentation (for example, for Modern C++, navigate to **Modules** > **RTI Connext API Reference** > **Domain Module** > **DomainParticipant** > **discovered_participant_subject_name**).

Note: These methods have different functionality when enabling the *Security Plugins*. Please refer to the *RTI Security Plugins User's Manual* for more information.

2.7 WaitSet and GuardCondition now implement AutoCloseable

WaitSet and GuardCondition objects require manual destruction, which was provided only as a **delete()** method. Starting in this release, these classes implement **java.lang.AutoCloseable**, which provides the **close()** method and allows for a simplified lifecycle management within a try block:

```
try (WaitSet waitset = new WaitSet()) {
    // Use waitset
    // ...
} // waitset deleted
```

2.8 InstanceHandle, condition types, and entity types are now hashable (Modern C++)

Template specializations of **std::hash** for the following types have been added:

- **dds::core::InstanceHandle**
- **dds::core::Entity** and its derived types (**DomainParticipant**, **Topic**, **DataReader**, etc.)
- **dds::core::cond::Condition** and its derived types (**GuardCondition**, **StatusCondition**, etc.)

This enhancement allows using these types as the keys of containers such as **std::unordered_set** and **std::unordered_map**.

2.9 New attribute for improving version compatibility of XML documents

This release introduces support for the must_interpret XML attribute. This attribute improves backward and forward compatibility of XML documents.

XML elements that have the must_interpret attribute set to false will not trigger a validation failure by the XML parser. Add must_interpret="false" to elements that are not supported in other *Connext* releases. When must_interpret is set to false, only the versions of *Connext* that understand these elements will interpret them. Others will ignore them when parsing the XML file.

If the must_interpret attribute is not specified, its default value is "true"—the XML parser will validate the element, as in previous releases.

> **Note:** Using the must_interpret attribute in your XML files breaks compatibility with versions of *Connext* prior to 7.0.0. For more information, read the *How the XML is Validated* section of the RTI Connext Core Libraries User's Manual.

3 Usability

3.1 DDS-XML format for Limited Bandwidth Endpoint Discovery Plugin

The Limited Bandwidth Endpoint Discovery plug-in (LBED) uses an XML file to statically specify the QoS (and other information, such as the *Topic* or type being used) of the endpoints that should be "discovered." However, this XML did not follow the OMG's DDS Consolidated XML Syntax (DDS-XML) (the way to specify the configuration, the labels, the schema, etc.), unlike other RTI elements such as USER_QOS_PROFILES.xml, XML-Based Application Creation, or *RTI System Designer*. This created a coexistence of two different ways to define DDS systems using XML in the RTI ecosystem: the standardized one used by most RTI tools and the one that only LBED used.

Since it is possible to represent the LBED configuration using DDS-XML, this release has updated the LBED plug-in to follow the DDS-XML standard. This way, if you already have a DDS-XML description of your system, it can be used directly for LBED, without needing to translate it to another schema.

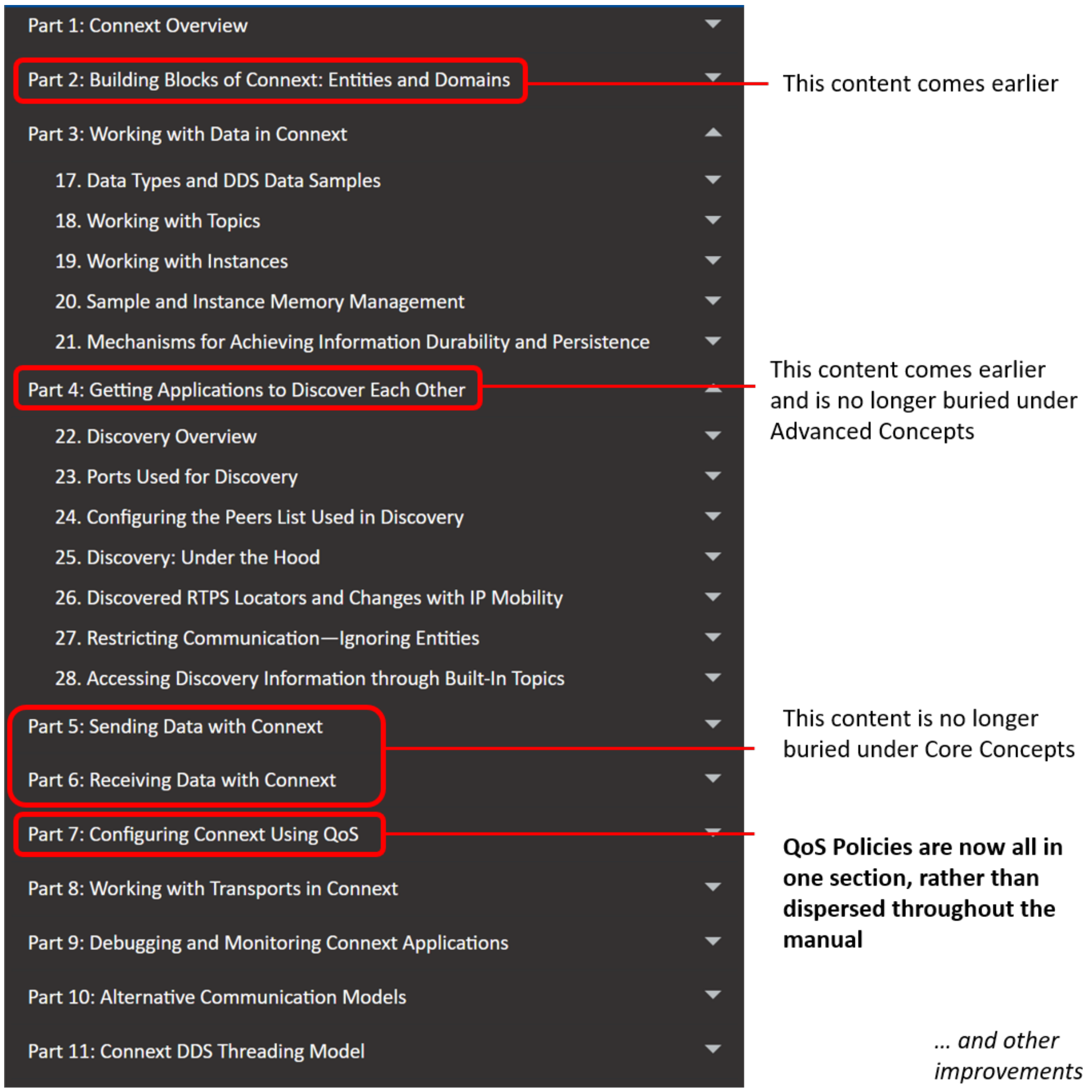
While improving the LBED XML configuration, other changes were made to improve the ease of use and LBED functionality, such as auto-determining the **rtps_object_id** of the endpoints when XML-Based Application Creation is used. These changes simplify the way in which LBED is enabled for a *DomainParticipant* and make additional XML configuration files optional. For more details about these enhancements, see the *RTI Limited Bandwidth Endpoint Discovery Plugin Release Notes* and the *RTI Limited Bandwidth Endpoint Discovery Plugin User's Manual*.

3.2 Reorganized Core Libraries User's Manual for easier browsing

The *RTI Connext Core Libraries User's Manual* has been reorganized to make it easier to find information. Some highlights:

- Previously, the Core Concepts and Advanced Concepts sections hid important chapters. For example, "Sending Data" and "Receiving Data" were placed under Core Concepts; "Discovery" and "Configuring QoS with XML" were placed under Advanced Concepts. These chapters are now moved "up" in the hierarchy and easier to find.
- Content such as "Discovery" and "Domains" occur earlier in the manual.
- All Quality of Service (QoS) information is now in one chapter, accessible from the top level of the table of contents. Previously, it was dispersed by entity throughout the manual.

Figure 1.2: Improved User's Manual Structure


Part 1: Connext Overview

Part 2: Building Blocks of Connext: Entities and Domains

Part 3: Working with Data in Connext

17. Data Types and DDS Data Samples
18. Working with Topics
19. Working with Instances
20. Sample and Instance Memory Management
21. Mechanisms for Achieving Information Durability and Persistence

Part 4: Getting Applications to Discover Each Other

22. Discovery Overview
23. Ports Used for Discovery
24. Configuring the Peers List Used in Discovery
25. Discovery: Under the Hood
26. Discovered RTPS Locators and Changes with IP Mobility
27. Restricting Communication—Ignoring Entities
28. Accessing Discovery Information through Built-In Topics

Part 5: Sending Data with Connext

Part 6: Receiving Data with Connext

Part 7: Configuring Connext Using QoS

Part 8: Working with Transports in Connext

Part 9: Debugging and Monitoring Connext Applications

Part 10: Alternative Communication Models

Part 11: Connext DDS Threading Model

This content comes earlier
This content comes earlier and is no longer buried under Advanced Concepts
This content is no longer buried under Core Concepts
QoS Policies are now all in one section, rather than dispersed throughout the manual
... and other improvements

3.3 Easier viewing of supported platforms

The *Core Libraries Release Notes* is now the one-stop place to see all supported platforms, for all products in *Connext Professional, Connext Secure*, and *Connext Anywhere*. There is one table for the compiler-dependent products, and one table for *Connext* tools and other standalone applications. Previously, you had to see each product's *Release Notes* to find out what's supported for each platform. Now you'll find that information in one document. See Supported Platforms for Compiler-Dependent Products, in the RTI Connext Core Libraries Release Notes.

3.4 System resources easier to manage with max_objects_per_thread now configured by Connext automatically

Previously, if you created multiple *DomainParticipants* in a single process, you would sometimes need to increase the value of **max_objects_per_thread**. In this release, *Connext* will automatically increase **max_objects_per_thread** as needed by the application. This is now the default behavior for **max_ objects_per_thread**.

This new behavior is useful because it is difficult to estimate the number of objects that will be required for a given application. Previously, customers used trial and error to arrive at a sufficient value. Now, *Connext* automatically increases **max_objects_per_thread** as needed.

If you are currently setting the value of **max_objects_per_thread** and wish to take advantage of the new default behavior, simply delete any code that sets the value.

In the very unlikely event that you need to set a limit on the number of thread-specific objects that are created (for example, you are running an extremely memory constrained application and are willing to risk runtime exceptions to save a small amount of memory), you can still set **max_objects_per_ thread**. Additionally, a new setting, **initial_objects_per_thread**, is available to control the initial capacity allocated for thread-specific objects.

Setting **initial_objects_per_thread** equal to **max_objects_per_thread** will cause all of the capacity to be allocated when *Connext* is initialized. This was the default behavior in previous releases.

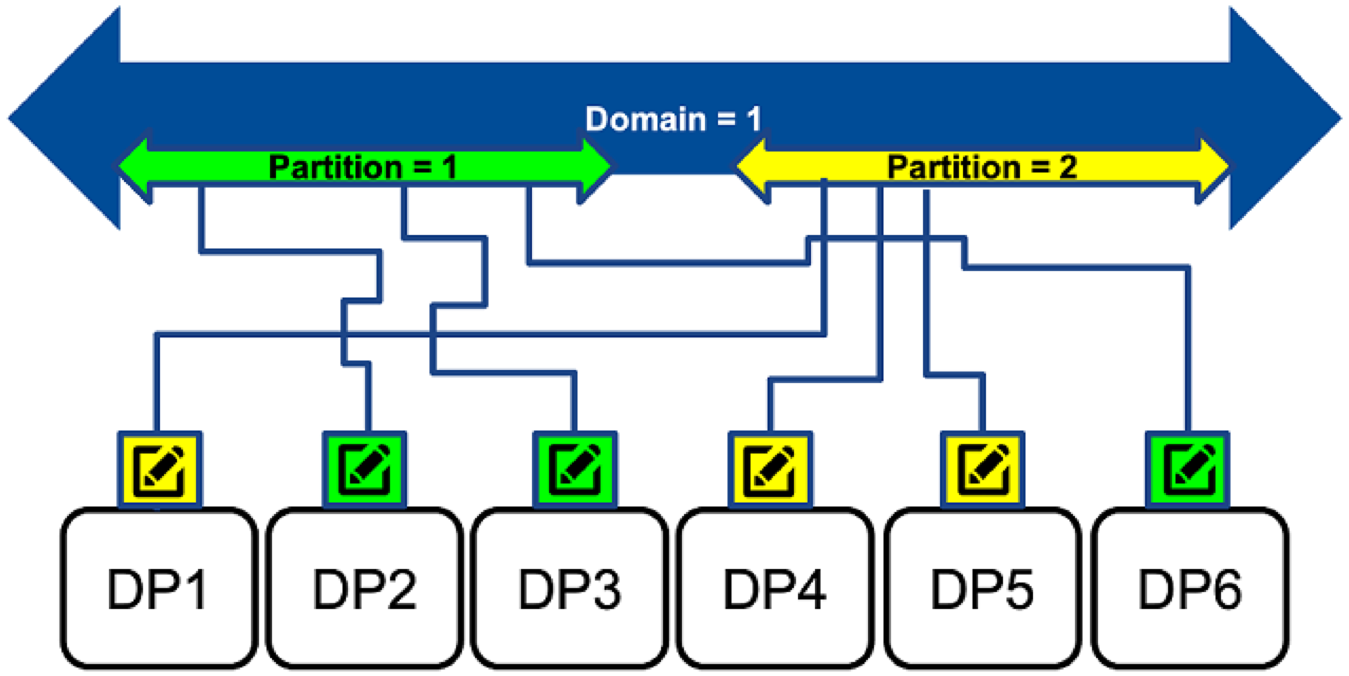
See the *SYSTEM_RESOURCE_LIMITS QoS Policy (DDS Extension)* section of the RTI Connext Core Libraries User's Manual for more information.

4 Scalability

4.1 Ability to partition groups of applications

Previously, the PARTITION QoS applied only to *Publishers* and *Subscribers*, controlling which *DataWriters* and *DataReader* could communicate (even if they matched with the same *Topic* and compatible QosPolicies). The PARTITION QoS policy now also applies to *DomainParticipants*. Now *DomainParticipants* with the same domain ID, domain tag, and at least one matching partition can communicate.

Figure 1.3: Partitions at the DomainParticipant Level



Partitioning at the *DomainParticipant* level can reduce the number of *DomainParticipants* that need to exchange endpoint discovery information. Partitioning at this level helps to reduce network, CPU, and memory utilization, because *DomainParticipants* without matching partitions will not exchange information about their *DataWriters* and *DataReaders*. Partitioning at the *DomainParticipant* level can be particularly useful in large, WAN, distributed systems (with thousands of participants) in which not all participants need to know about each other at any given time.

DomainParticipant partitions work just like *Publisher* and *Subscriber* partitions, except they apply at the *DomainParticipant* level.

As opposed to domain tags, *DomainParticipant* partitions can be changed dynamically. In addition, a *DomainParticipant* can be part of multiple partitions at once, and you can use regular expressions to match partitions.

DomainParticipant partitions are independent of *Publisher* and *Subscriber* partitions. You can use both features independently or in combination to provide the right level of isolation.

See the *PARTITION QosPolicy* section of the RTI Connext Core Libraries User's Manual for more information.

5 Bandwidth Sensitivity

5.1 Reduce network traffic for bandwidth-sensitive applications by disabling ServiceRequest channel

The ServiceRequest channel is a builtin channel that supports the TopicQuery and locator reachability features. This channel generates network traffic and creates entities that consume memory and CPU cycles.

If you are not using the TopicQuery or locator reachability features and would like to avoid incremental network traffic, you can disable the creation of the ServiceRequest channel with a new *DomainParticipant* QoS setting: **enabled_builtin_channels** (part of the DISCOVERY_CONFIQ QoS Policy). Please refer to the API Reference manual for your language or the *DISCOVERY_CONFIG QosPolicy (DDS Extension)* section of the RTI Connext Core Libraries User's Manual for detailed documentation on **enabled_builtin_channels**.

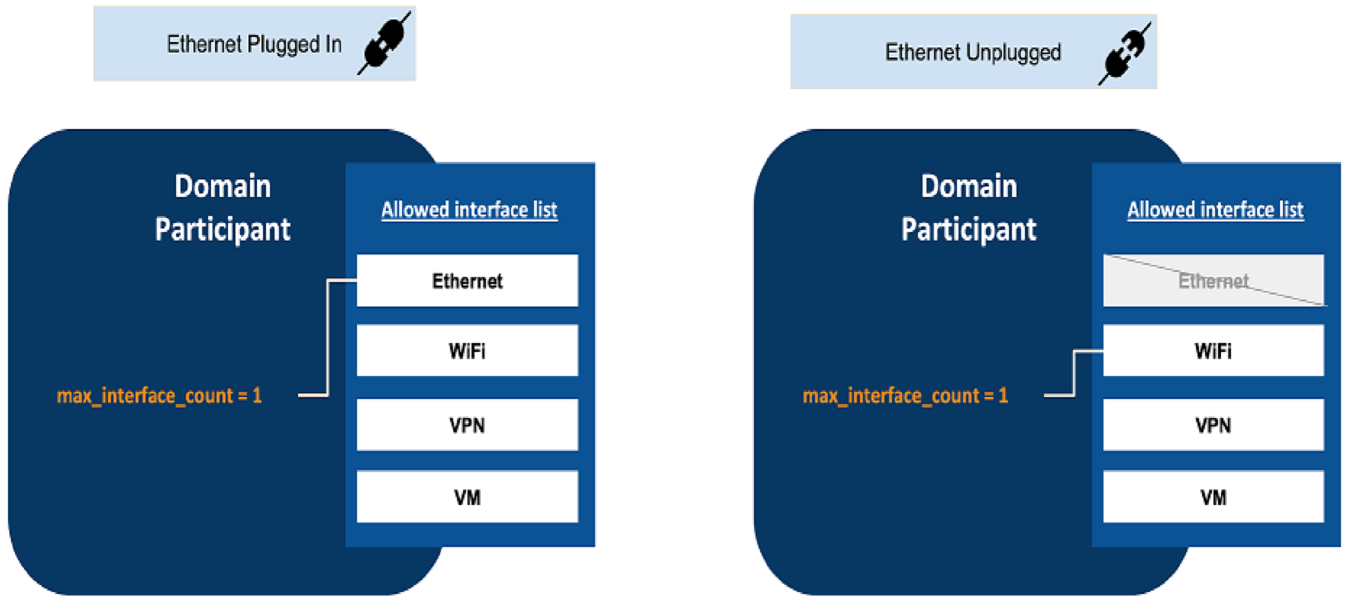
Note: Disabling the ServiceRequest channel will disable the TopicQuery and locator reachability features. Therefore, errors will be generated if you create a TopicQuery, enable TopicQuery dispatch, or enable locator reachability while the ServiceRequest channel is disabled.

5.2 Reduce network bandwidth utilization by allowing Connext to limit allowed interfaces even further

Connection availability can be unpredictable in some environments. As a result, devices usually provide multiple connections. Previously, a *DomainParticipant* announced and received data over all up-and-running interfaces in the allowable interfaces list, which could sometimes result in data duplication and poor use of network bandwidth.

In this release, a *DomainParticipant* can now be configured to receive data over preferred interfaces only, by setting a new property, **max_interface_count**, which can be used in all IP-based transports. This property limits the number of network interface locators to be included in the *DomainParticipant's* announcement, prioritizing whichever interface(s) are specified first (in a left-to-right manner) in the **allow_interfaces_list** for the transport.

Figure 1.4: Interface Selection Using max_interface_count



For example, you may have one wired and one wireless interface up and running. Receiving data over the wireless connection is only desired if no wired connectivity is available (for example, when the device is undocked). If **max_interface_count** is set to 1, the *DomainParticipant* will receive data over the interface you list first—for example, the wired interface. That way, when both interfaces are up and running, you will receive data only over the wired interface. If the wired interface is not in use (for example, the device is undocked), then you will receive data only over the next available up-and-running interface in your **allow_interfaces_list**, which would be the wireless interface.

The **max_interface_count** property also affects multicast traffic by limiting the interfaces over which a *DomainParticipant* sends multicast traffic.

The **max_interface_count** setting does not consider end-to-end connectivity to select interfaces. The decision is based purely on whether interfaces are up or down in a node. Therefore, this feature is not intended to be used in the following scenarios:

- A *DomainParticipant* is not reachable by other *DomainParticipants* in all the interfaces in the **allow_interfaces_list**. This could occur if the *DomainParticipant* is in different subnets, and some of these subnets cannot be reached by other *DomainParticipants*.
- End-to-end connectivity issues lead to situations in which the interfaces selected after applying **max_interface_count** cannot be reached by other *DomainParticipants*.

See information about the new **max_interface_count** field in the documentation for your transport, such as in the *Setting Builtin Transport Properties with the PropertyQosPolicy* section of the RTI Connext Core Libraries User's Manual.

5.3 More efficient bandwidth utilization for configurations with small transport message_size_max

Connext adds protocol information to every RTPS message it sends out. Before release 7.0.0, *Connext* reserved 512 bytes for the protocol information out-the-box regardless of whether the bytes were used.

Reducing the *Connext* transport MTU (<message_size_max>) led to a small payload utilization ratio per RTPS packet. For example, if you reduced the transport MTU to 1400 bytes to disable IP fragmentation, the maximum number of bytes per RTPS packet was 888.

To improve bandwidth utilization, *Connext* offered a property, **dds.participant.protocol.rtps_overhead**, that you could use to adjust the protocol overhead to a value smaller than the default 512. However, coming out with a good value for the property was not easy. In addition, if the value was too small, *Connext* could not send samples.

Starting with release 7.0.0, the calculation of the RTPS protocol overhead is automatically done by *Connext* per message, leading to more efficient bandwidth utilization. The property **dds.participant.protocol.rtps_overhead** has been deprecated, since there is no need to use it anymore.

6 Security Improvements

This section describes security-related improvements made in the Core Libraries. For a complete list of new features in the *RTI Security Plugins*, see the *RTI Security Plugins Release Notes*.

6.1 Troubleshoot security issues faster using new SECURITY logging category

A new logging category, NDDS_CONFIG_LOG_CATEGORY_SECURITY, is introduced in this release. It enables you to filter and more easily trace *Security Plugins*-related code operations.

Along with the new SECURITY logging category, the following enhancements have been made:

- More activity context for *Security Plugins*-related messages, to provide higher-level context in which the problem or event occurred (such as CREATE DP).
- Improved messages, to provide more detail on lower-level events (such as failure to allocate 10 bytes of memory).

For information on filtering messages by category, see the *Configuring Connext Logging* section of the RTI Connext Core Libraries User's Manual.

Note that the *Security Plugins* still have their own logging mechanism. All messages printed by the *Security Plugins* now also use the SECURITY category. See the "Security Events and Logging" section in the *RTI Security Plugins User's Manual* for more information.

6.2 Secure compressed batches of data using Security Plugins

Previously, the combination of compression, batching, and data protection via *RTI Security Plugins* was not supported and resulted in a *DataWriter* creation error. Now, this combination is supported, because data protection is applied to the entire batch. The batch will first be compressed, and then the compressed batch will be protected. See "Data Compression" in the *DATA_REPRESENTATION QosPolicy* section of the RTI Connext Core Libraries User's Manual for more details on compression.

7 Logging Improvements

7.1 Determine severity of log messages using log level prefixes

Previously, log messages did not include their severity. For example:

```
PRESTypePluginDefaultEndpointData_createWriterPool:!create writer buffer pool
```

It was hard to determine if a log message was an error or warning, or just debugging information.

The improved log message format can be seen in the following examples:

```
LOCAL [0x0101A25C,0x4C0B3571,0x25152FD6:0x000001C1\{N=fooParticipant,D=0}|CREATE DP|ENABLE]
COMMENDActiveFacade_addReceiverThreadEA:thread count ref count 2
```

```
WARNING [0x0101A25C,0x4C0B3571,0x25152FD6:0x000001C1\{N=fooParticipant,D=0}|CREATE
DP|ENABLE] NDDS_Transport_UDPv4_Socket_bind_with_ip:0X1CF2 in use
```

```
ERROR [CREATE DP] DDS_DomainParticipantFactory_create_participant_disabledI:ERROR: Bad
parameter: qos
```

For information on log levels and formatting log messages, see the *Configuring Connext Logging* section of the RTI Connext Core Libraries User's Manual.

7.2 Improved logging in specific scenarios

7.2.1 Changes in liveliness

Log messages for liveliness change scenarios have been updated to show information about the remote and local entity. For example:

```
[0x010117A7,0x1685DF88,0x8299F0FF:0x80000002
Unknown macro: {E=DW,D=2}
|LIVELINESS CHANGE] PRESPsService_writerActivityListenerOnRemoteReaderInactive:local writer
liveliness change to BECAME_INACTIVE in remote reader
0x01018923,0x66B78325,0x0E38DE7E:0x80000007
```

These messages will appear when logging at the NDDS_CONFIG_LOG_LEVEL_STATUS_LOCAL verbosity level or higher.

7.2.2 Inconsistent PROPERTY QoS policies

If you exceed the maximum number of properties or maximum string length for a property in the PROPERTY QoS Policy, you will now see an error message with the limit and current value, to help determine the length or number of properties you need to change. An example new error message looks like the following:

```
DDS_PropertyQosPolicy_is_consistentI:inconsistent QoS policies: number of properties in the
property QoS policy (2) and DDS_DomainParticipantQos.resource_limits.participant_property_
list_max_length (0). By default, the property QoS policy in a participant QoS is populated
with some system properties. Delete these properties or increase the resource limit DDS_
DomainParticipantQos.resource_limits.participant_property_list_max_length
```

7.2.3 Discovery of remote entity using non-addressable locator

If a remote application tries to communicate with a local application that has a different transport configuration, it is going to log a message indicating that a remote entity that is using a non-addressable locator has been discovered. This will also happen if multicast is enabled on the remote, but not on the local, participant. For example:

```
COMMENDSrWriterService_assertRemoteReader:Discovered remote reader with GUID
0XA0AC1E9,0X5838,0X1,0X20087 using a non-addressable multicast locator.
This can occur if multicast is not enabled in the local participant.
See https://community.rti.com/kb/what-does-cant-reach-locator-error-message-mean for
additional info.
can't reach:
locator: udpv4://239.255.0.1:29900
aliasList: udpv4,shmem
```

This type of message will not be printed as an error anymore, because transport configuration can be different between participants; this scenario is expected. This message will now be logged at the NDDS_CONFIG_LOG_LEVEL_STATUS_LOCAL level. *Connext*, however, will log a warning if there are no reachable locators for a remote entity, with the following message:

```
COMMENDBeWriterService_assertRemoteReader:The remote reader with GUID
0x010166E0,0xFBD6F2FC,0x1F354248:0x80000004 has no addressable locators.
A locator is unreachable if its transport is not installed and/or enabled in the local
participant.
See https://community.rti.com/kb/what-does-cant-reach-locator-error-message-mean for
additional info.
```

7.2.4 Improved logging in write code path

New log messages have replaced unclear, old ones, and additional messages have been added where context was missing, for messages in the write code path. These improvements make it easier to understand a failure and simplify the debugging process for write code path scenarios. Some examples are the following:

BEFORE

```
ERROR [DELETE DP|LC:DISC]WriterHistoryMemoryPlugin_addSample:out of order
```

AFTER

```
ERROR [0x0101C93E,0x9FB304CC,0xD6BEFB42:0x000001C1{D=0}|CREATE DP|ENABLE|ADD TO WRITER
QUEUE|LC:DISC]WriterHistoryMemoryPlugin_addSample:OUT OF ORDER | Current timestamp
(2022-01-
11 02:14:56) and last timestamp (2022-01-11 18:27:40)
```

BEFORE

```
ERROR [DELETE DP|LC:DISC]WriterHistoryMemoryPlugin_addInstance:writer history full
```

AFTER

```
ERROR [0x0101F77B,0x53D5D4C7,0x48D3DFDE:0x000001C1{D=0}|CREATE DP|ENABLE|ADD TO WRITER
QUEUE|LC:DISC]WriterHistoryMemoryPlugin_addInstance:OUT OF RESOURCES | Exceeded max
number
of instance (writer_qos.resource_limits.max_instances). Number of instance (1) and
maximum
(1)
```

7.2.5 Serialization error now displays both message_size_max and actual message size values for easier comparison

When *Connext* exceeds the maximum size of the serialization buffer, the error now shows the current value along with the limit value, so that you can know the status of the buffer and take action on the issue more clearly. An example of the new error message is as follows:

```
ERROR [0x0101707E,0xF19D1787,0x0CECDBF7:0x000001C1\{D=0}\|CREATE
DP\|ENABLE\|LC:DISC]MIGGenerator_addData:add data failed. The most likely cause is that the
message size max ('950') (for at least one of the installed transports) is too small for
propagating the participant discovery information, with message size (956).
```

7.2.6 One, unified message for all "open file" failure log messages

There were many different versions of the error message logged when a file could not be opened. These error messages have been unified into one message, which prints the name of the file, in the following format:

```
Failed to open file 'example.xml'
```

7.3 Changes to log message verbosity

7.3.1 "No initial peers" log message moved to higher verbosity level

If you enabled a *DomainParticipant* with an empty list of initial peers, you received the following warning message:

```
[D0122|ENABLE] DDS_DomainParticipantDiscovery_enableI:no peer locators for: peer descriptor
(s) = "", transports = "", enabled_transports = ""
```

You will no longer receive a warning when declaring a *DomainParticipant* with no initial peers, because this is a valid scenario. Now the empty list of initial peers scenario is reported at the NDDS_CONFIG_LOG_LEVEL_STATUS_LOCAL verbosity level.

For information on log levels, see the *Configuring Connext Logging* section of the RTI Connext Core Libraries User's Manual.

7.3.2 Warning message about USER_QOS_PROFILES.xml moved to higher verbosity level

Previously, when using the NDDS_CONFIG_LOG_VERBOSITY_WARNING log level, you would see the following in your list of messages:

```
DDS_DomainParticipantFactory_initializeI:Welcome to NDDS
NDDSCORE_BUILD_6.0.1.0_20191115T195316Z_RTI_REL
NDDSC_BUILD_6.0.1.0_20191115T195316Z_RTI_REL
NDDSCPP_BUILD_6.0.1.0_20191115T195316Z_RTI_REL
DDS_XMLParser_parse_from_file:Loading : USER_QOS_PROFILES.xml
```

This information, about the version number and Qos profile XML loading, is not considered a warning in the application. Therefore, the verbosity of this message has been upgraded to a higher verbosity, NDDS_CONFIG_LOG_VERBOSITY_STATUS_LOCAL. Now, you will only see this message when using the NDDS_CONFIG_LOG_VERBOSITY_STATUS_LOCAL log level.

See the *Configuring Connext Logging* section of the RTI Connext Core Libraries User's Manual for more information on verbosity levels.

7.4 Changes to Activity Context in log messages

In release 6.1.0, *Connext* added context (activity and resource information) to log messages. For example, the following log message corresponds to an error during the write operation (activity) that includes contextual information about the associated *DataWriter* and *Topic* (resources):

```
[0X1019D1D,0XBD6B47B0,0XA6C11F6B:0X80004202{E=DW,T=Example Stock,C=Stock,D=1}|WRITE]
WriterHistoryMemoryPlugin_addSample:instance not found
```

For more information about Activity Context, see the *Format of Logged Messages* section of the RTI Connext Core Libraries User's Manual.

The following enhancements have been made to the Activity Context of log messages.

7.4.1 Activity Context in messages shows full name of resource attribute (such as Topic) instead of first letter (such as T)

Previously, the Activity Context showed only the first letter of resource attributes: T for *Topic*, C for Type, E for *Entity*, D for Domain, N for Name, and I for message ID. Now, the Activity Context shows the full name of the resource attribute: *Topic*, Type, *Entity*, Domain, Name, or MessageKind (instead of I for message ID).

For example, a log message like the following:

```
[0X101A76B,0X79E5D71,0X50EE914:0X1C1:0X80000088{E=Pu,D=1}|CREATE Writer WITH TOPIC
myTopicName]
```

Will now look like this:

```
[0X101A76B,0X79E5D71,0X50EE914:0X1C1:0X80000088{Entity=Pu,Domain=1}|CREATE Writer WITH TOPIC
myTopicName]
```

7.4.2 Activity Context now fully states what kind of message is being received

Activity Context now spells out what kind of message is being received. Instead of a "I=" to indicate a message ID, it now states that a MessageKind is a "DATA", "HEARTBEAT", "GAP", or other message kind.

For example, before the Activity Context was as follows:

```
[0xC0A87A01,0x00007BFC,0x00000001:0x000201C4{Entity=DR,I=21}
```

Now, it is as follows:

```
[0xC0A87A01,0x00007BFC,0x00000001:0x000201C4{Entity=DR,MessageKind=DATA}
```

7.4.3 Identify source of event execution message more easily with added resources and activities information

Messages logged during event execution now also include contextual information from the original posting thread. For example, imagine that *Connext* fails to send an ACK, described in the following error message: “!send periodic ACK”. Now, *Connext* provides additional contextual information, which includes information about the action the original thread was executing “[ASSERT REMOTE DW]” and the associated local *DataReader* and remote *DataWriter* GUID. The message looks like this:

```
[0x0101FECD,0xE9F4860F,0x1E657387:0x000003C7|ASSERT REMOTE DW
0x010143CC,0xF977283C,0xE705F81F:0x000003C2] COMMENDSrReaderService_onAckPeriodicEvent:!send
periodic ACK
```

7.5 View total heap memory usage by the middleware using new field in Heap Monitoring

The Heap Monitoring utility enables you to measure the amount of heap memory in bytes used by the middleware. You can find this information in the "Current application heap usage" field in the heap memory snapshot.

This value, however, does not include overhead memory allocations that are used by the Heap Monitoring utility. It shows the heap usage when Heap Monitoring is disabled and does not reflect the actual amount of memory that has been allocated by the middleware.

This missing information is now accounted for in a new field, "Approximate total heap usage," in the heap memory snapshot. This field measures the amount of heap memory in bytes used by the middleware, including overhead allocations from the Heap Monitoring utility.

For more information, see the API Reference HTML documentation (for example, in Modern C++, select **Modules > RTI Connext API Reference > General Utilities > Heap Monitoring > take_snapshot**).

8 DDS Spy and DDS Ping Improvements

8.1 Debug Connext applications from command line faster with new DDS Spy output format

DDS Spy's output format has been improved and simplified to enable a faster debugging experience. The following improvements have been made:

- **Timestamp**. Previously, the timestamp was a counter (for example "1639401977.271911"). Now, it is printed in Coordinated Universal Time (UTC). For example: "14:59:16". Milliseconds are no longer included, to simplify output.
- **Info**. Previously, Info was three digits describing each sample. For example, "R +N" meant another *DataReader* was discovered. Now, *DDS Spy* provides a full description of each sample, such as "New reader".
- **Source**. Previously, the Src HostId was printed in hexadecimal and described the entity (for example "C0A82BDF"). Now, the source IP is printed, so that you know the source of the data. For example: "from 192.168.43.223".
- The rest of the information is printed following the key-value format. For example: "topic="Example test" type="test"".

For more information on these and the following features, see the RTI DDS Spy User's Manual.

8.1.1 Get a statistical summary of DDS Spy’s session when you exit Spy

Now, when you perform "Ctrl+c" or the application times out, *DDS Spy* prints how many endpoints have been discovered and how many samples have been received, by *Topic*. For example:

```
---- Statistics ----
Discovered 1 DataWriters and 1 DataReaders
Received samples (Data, Dispose, NoWriters):
8, 0, 0     (Topic="Example test"  Type="test")
```

8.1.2 Select from two modes to print samples in DDS Spy: Plain and Compact

There are now the following options for displaying the sample:

- PLAIN (**-printSample PLAIN**). This is the default option. It pretty-prints sample information for best readability.

```
12:35:24 New data          from 192.168.43.223  : topic="Circle" type="ShapeType"
color: "BLUE"
x: 53
y: 190
shapesize: 30
fillKind: SOLID_FILL
angle: 0
```

- COMPACT (**-printSample COMPACT**): Prints sample information in a single line using a JSON format.

```
12:38:15 New data          from 192.168.43.223  : topic="Circle" type="ShapeType"
sample={"color":"BLUE","x":202,"y":175,"shapesize":30,"fillKind":"SOLID_
FILL","angle":0}
```

8.1.3 View discovered entity names in DDS Spy using new option

You can now ask *DDS Spy* to print the entity name of the discovered entities and the name of the *DataWriter* sending the data using the option **-showEntityName**:

```
13:26:17 New reader        from 192.168.43.223  : topic="Example test" type="test"  entity_
name="roleName:testDataReader"
```

8.1.4 View partitions of discovered entities in DDS Spy using new option

You can now ask *DDS Spy* to print the partition of the discovered entities and the partition of the *DataWriter* using the option **-showPartition**:

```
13:26:17 New reader        from 192.168.43.223  : topic="Example test" type="test"
partition="A, B, C"
13:26:15 New writer        from 192.168.43.223  : topic="Example test"
type="test"partition="A, E, I"
15:41:09 New data          from 192.168.43.223  : topic="Example test" type="test"
type="test"partition="A, E, I"
15:41:10 New data          from 192.168.43.223  : topic="Example test" type="test"
type="test"partition="A, E, I"
```

8.2 Troubleshoot in DDS Spy using three modes: discovery, user data, or everything

By default, *DDS Spy* prints both discovery and user data together; however, now *DDS Spy* provides you the choice of printing either discovery or user data, by specifying a mode:

- **-mode USER**
  - Skip discovery data.
  - Force **-printSample** argument
- **-mode DISC**
  - Skip user data.
  - Force **-showEntityName**, **-showHandle**, and **-showPartition** arguments.

See information about "Discovery vs. User modes" in the RTI DDS Spy User's Manual.

8.3 View updated information in DDS Spy as an endpoint is modified

Previously, if you updated the value of an endpoint, that data was not reflected in *DDS Spy*.

In this example, the *DataWriter* changed its partition from "P1" to "P2" and the IP from "192.168.42.107" to "192.168.55.133", but previously none of those changes was shown in the output:

```
13:29:20 New writer        from 192.168.42.107   : topic="Example test" type="test"
partition="P1"
13:29:22 New data          from 192.168.42.107   : topic="Example test" type="test"
partition="P1"
13:29:23 Updated writer    from 192.168.42.107   : topic="Example test" type="test"
partition="P1"
13:29:24 New data          from 192.168.42.107   : topic="Example test" type="test"
partition="P1"
```

Now when an endpoint updates its data, the change is visible in the *DDS Spy* output, as shown in this example:

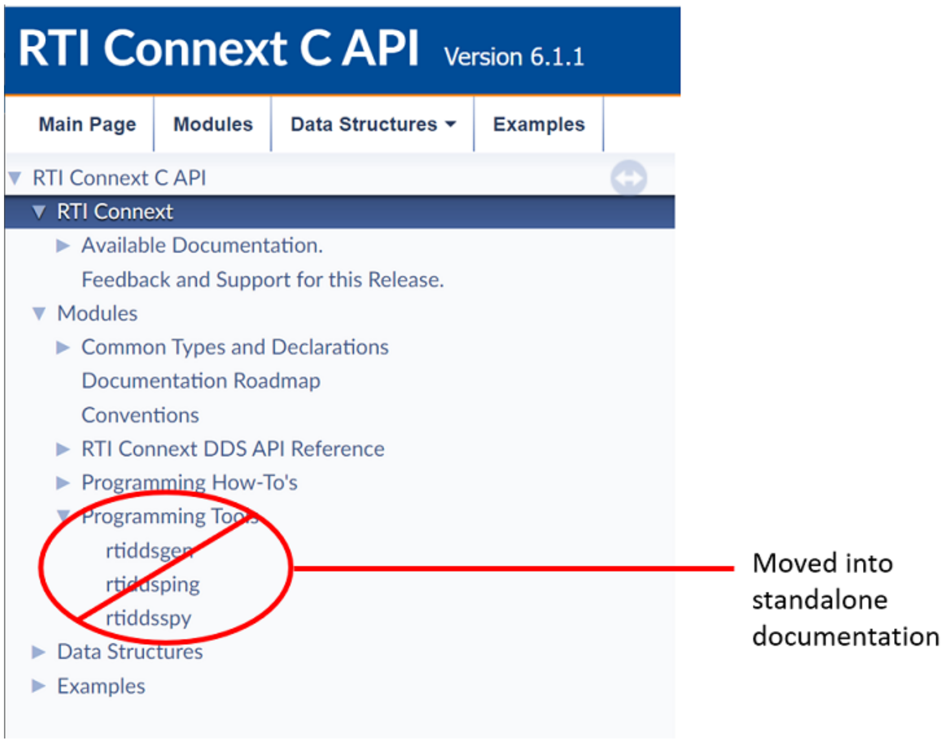
```
13:29:20 New writer        from 192.168.42.107   : topic="Example test" type="test"
partition="P1"
13:29:22 New data          from 192.168.42.107   : topic="Example test" type="test"
partition="P1"
13:29:23 Updated writer    from 192.168.55.133   : topic="Example test" type="test"
partition="P2"
13:29:24 New data          from 192.168.55.133   : topic="Example test" type="test"
partition="P2"
```

8.4 Standalone documentation for DDS Spy and DDS Ping

As part of the improvements in *DDS Spy*, the product's documentation has been moved out of the API Reference HTML documentation and into its own standalone document, the RTI DDS Spy User's Manual. This documentation contains more examples, and makes command and output descriptions easier to find and read. *DDS Ping's* documentation has also been moved out of the API Reference HTML documentation and improved; see the RTI DDS Ping User's Manual.

For consistency, information formerly in the rtiddsgen section of the API Reference HTML documentation's "Programming Tools" section has also been moved to the RTI Code Generator User's Manual.

Figure 1.5: Moved Documentation



9 Platform and Build Changes

9.1 Dynamically Load Monitoring Library and Security Plugins on VxWorks

Connext has the capability to enable the Monitoring Library and the *Security Plugins* using QoS settings without the need to recompile an application. This release adds support for these features on VxWorks platforms. See the *Method 1—Change the Participant QoS to Automatically Load the Dynamic Monitoring Library* section of the RTI Connext Core Libraries User's Manual and information about Dynamic linking in the "Linking Applications with the Security Plugins" section in the *RTI Security Plugins User's Manual* for details on the QoS properties used to enable these features.

9.2 New Supported Platforms

This release adds support for the following platforms, compared to release 6.1.1:

Table 1 New Platforms

OS	OS Version	CPU	Toolchain	RTI Architecture
Linux	Ubuntu 22.04 LTS	x64	gcc 7.3.0	x64Linux4gcc7.3.0
	Ubuntu 22.04 LTS	Arm v8	gcc 7.3.0	armv8Linux4gcc7.3.0
Windows	Windows 11	x64	VS 2022	x64Win64VS2017
macOS	macOS 12 Validated using the same libraries as macOS 10.x and 11	x64	clang 13.0	x64Darwin17clang9.0
	macOS 12 (target only)	Arm v8	clang 13.0	arm64Darwin20clang12.0
VxWorks	VxWorks 21.11	x64	llvm 12.0.1.1	x64Vx21.11llvm12.0.1.1 x64Vx21.11llvm12.0.1.1_rtp

10 Changes to Defaults

10.1 shutdown_cleanup_period default changed from 1s to 10ms

The default value of the **shutdown_cleanup_period** in the DATABASE QoS Policy has changed from 1 second to 10 milliseconds to speed up participant deletion times.

10.2 max_objects_per_thread default changed from 2048 to 261120

The default value for **max_objects_per_thread** in the SYSTEM_RESOURCE_LIMITS QoS Policy has changed from 2048 to 261120. See 3.4 System resources easier to manage with max_objects_per_thread now configured by Connext automatically on page 12 for more information.

10.3 min_initial_participant_announcement_period default changed from 1 second to 10 milliseconds

The default value for **min_initial_participant_announcement_period** in the DISCOVERY_CONFIG QoS Policy has changed from 1 second to 10 milliseconds.

Before, both the **min_initial_participant_announcement_period** and **max_initial_participant_announcement_period** default values were 1 second, which did not provide a default range. Now, the change of **min_initial_participant_announcement_period** to 10 milliseconds provides a range by default. As explained in the current version of the *RTI Connext Core Libraries User's Manual*, the randomness within the range set by these values reduces the chances of a network collision when multiple participants are started at the same time.

11 Performance Improvements

11.1 Use of TopicQueries or ContentFilteredTopics may incur lower overhead

This release introduces a reliability protocol improvement that may reduce bandwidth consumption in scenarios where you create a lot of TopicQueries and/or ContentFilteredTopics.

The improvement reduces the number of RTPS GAP sub-messages that are sent in response to a NACK message requesting:

- Samples that do not pass the *DataReader's* ContentFilteredTopic expression.
- TopicQuery samples that are not part of the response to a *DataReader's* TopicQuery—that is, samples directed to TopicQueries from other *DataReaders*.

11.2 Improved performance when calling build_data and get_loan FlatData APIs at the same time that samples are published

There was a concurrency issue in the **build_data** and **get_loan** FlatData™ APIs. The calls to these methods would block if the *DataWriter* was publishing or repairing a sample. This may have led to significant concurrency issues.

For example, when using asynchronous publication, which typically is required with large data scenarios, you may have noticed that the time required by the **build_data** and **get_loan** FlatData APIs had a high jitter. This was because **build_data/get_loan** were taking the same lock as the asynchronous publisher publishing the data. This problem has been solved.

Note that this concurrency improvement does not apply to scenarios in which FlatData is used in combination with ZeroCopy.

11.3 Improved performance for unbounded sequences of primitive types (Modern C++)

This release includes an optimization that avoids unnecessary initialization of the memory of a **std::vector<T>** when T is a primitive type. This may result in improved performance for applications that use unbounded sequences in their user data types.

12 Deprecations and Removals

This section describes products, features, and platforms that are *deprecated* or *removed* starting in release 7.0.0.

Deprecated means that the item is still supported in this release, but will be removed in a future release. *Removed* means that the item is discontinued or no longer supported. By specifying that an item is deprecated in this release, RTI is hereby providing customer notice that RTI reserves the right after one year from the date of this release and, with or without further notice, to immediately terminate maintenance (including without limitation, providing updates and upgrades) for the item, and no longer support the item, in a future release.

This section serves as notice under the Real-Time Innovations, Inc. Maintenance Policy #4220 and/or any other agreements by and between RTI and customer regarding maintenance and support of RTI's software.

12.1 Legacy C# / .NET binding removed in this release

A new C# language binding for .NET 5 and .NET Standard 2.0 has been available since release 6.1.0, replacing the previous binding. The previous binding was deprecated in that release and has now been removed as of release 7.0.0. See the RTI Code Generator Release Notes for more information.

12.2 Support for pre-C++11 compilers removed for Modern C++ API

Support for pre-C++11 compilers has been removed for the Modern C++ API in release 7.0.0. The Modern C++ API now requires C++11 compilers (or newer). The Traditional C++ API supports C++98 compilers (or newer).

Likewise, the *RTI Code Generator* option **-language C++03** has also been removed as of release 7.0.0. (It was deprecated starting in release 6.1.0.) For the Modern C++ API, use **-language C++11**; for the Traditional C++ API, use **-language C++98**. See the RTI Code Generator Release Notes for more information.

Applications that used types generated with **-language C++03** may need minor updates for types generated with **-language C++11**; see Modern C++ API now maps enums to enum class in the Migration Guide.

12.3 Old command-line options deprecated and removed in RTI DDS Spy

Starting in release 6.1.1, the following command-line options in *DDS Spy* were deprecated, since they were only needed for backward compatibility with older releases:

- -use510CompatibleLocatorKind
- -use43LargeDataFormat
- -use530dynamicData

Of these, the following are now removed as of release 7.0.0:

- -use510CompatibleLocatorKind
- -use43LargeDataFormat

Although deprecated, -use530dynamicData is not yet removed, but may be removed in a future release.

12.4 Durable writer history, durable reader state, and Persistence Service no longer support external databases

Support for external databases was deprecated starting in release 6.1.1. In release 7.0.0, support for external databases (e.g., MySQL) is removed from the following features and components:

- Durable writer history
- Durable reader state
- *Persistence Service*

In *Persistence Service*, use the <filesystem> tag instead of the <external_database> tag to store samples on disk.

Support for durable writer history and durable reader state has been temporarily disabled in this release because these features were only supported with external relational databases. RTI will provide a file-based storage option for durable writer history and durable reader state in a future release. Contact RTI Support at support@rti.com for additional information regarding durable writer history and durable reader state.

12.5 RTI Secure WAN Transport removed in this release

Secure WAN Transport was deprecated starting with release 6.1.1 and is no longer supported as of release 7.0.0. You should use *RTI Real-Time WAN Transport* instead. See the *RTI Real-Time WAN* part of the RTI Connext Core Libraries User's Manual for more information.

12.6 RTI Prototyper removed in this release

Prototyper was deprecated starting in release 6.1.0 and has been removed as of release 7.0.0.

12.7 RTI Spreadsheet Add-in for Microsoft Excel removed in this release

Spreadsheet Add-in for Microsoft Excel was deprecated starting in release 6.1.0. As of release 7.0.0, it is no longer supported.

12.8 RTI Connector for Python deprecated in this release

With the introduction of the *Connext Python API*, *Connector for Python* is deprecated. It will be removed in a future release when the Python API is fully supported. We encourage you to try the Python API (see 2.1 Connext Python API adds support for user data types and code generation (experimental) on page 6).

12.9 Removed Platforms

The following platforms are no longer supported, starting with release 7.0.0:

- Linux platforms:
    - CentOS 6.x
    - Raspbian Wheezy 7
    - Red Hat Enterprise Linux 6.x
    - SUSE 12 and 15
    - Ubuntu 16.04 LTS
- INTEGRITY 11.0.4 on x86, 10.0.2
- QNX 6.x
- VxWorks 6.x, 7 SR0510
- Windows 8 and 8.1, Windows Server 2012
- Visual Studio 2012 and 2013

12.10 Deprecated Platforms

macOS 10.13 is supported in release 7.0.0, but will be removed in a future LTS release.

In the future LTS release, 32-bit Visual Studio 2015 and 2017 will be transitioned to custom-supported targets. (These are not supported in 7.0.0.)

12.11 Deprecated rtps_overhead Property

The property **dds.participant.protocol.rtps_overhead** has been deprecated, since there is no need to use it anymore. See 5.3 More efficient bandwidth utilization for configurations with small transport message_size_max on page 16 for an explanation.

13 Product Availability Changes

13.1 RTI Queuing Service

Queuing Service is not included in release 7.0.0. In a future LTS release, it will be available as an experimental feature in RTI Labs, https://www.rti.com/developers/rti-labs.

If you already have *Queuing Service* from another release in which it is fully supported, you can continue using it as such in that release. See *Experimental Features* in the RTI Connext Core Libraries Release Notes for information on RTI experimental products and features.

13.2 RTI Database Integration Service

Database Integration Service is not included in release 7.0.0.

13.3 RTI Limited Bandwidth Endpoint Discovery Plugin

In release 7.0.0, the new and improved *RTI Limited Bandwidth Endpoint Discovery Plugin* (LBED) (see 3.1 DDS-XML format for Limited Bandwidth Endpoint Discovery Plugin on page 9) is now included with the purchases of some bundles, including *Connext Professional*. It is still installed separately. See the *RTI Limited Bandwidth Endpoint Discovery Plugin Installation Guide*.

If you need to use the LBED plugin in production, it is available in *Connext* 6.1.1 LTS or earlier with *RTI Limited bandwidth Plugins* (an add-on product). The LBED enhancements in release 7.0.0 (see 3.1 DDS-XML format for Limited Bandwidth Endpoint Discovery Plugin on page 9) will be available in a future official product release.

14 Third-Party Software Upgrades

The following third-party software used by the Core Libraries has been upgraded:

Third-Party Software	Previous Version	Current Version
Expat	2.4.4	2.4.8
Zlib	1.2.11	1.2.12

Some of these upgrades may fix potential vulnerabilities. See *Fixes Related to Vulnerabilities*, in "What's Fixed in 7.0.0," in the RTI Core Libraries Release Notes.

For other third-party upgrades, see other products' release notes.

For information on third-party software used by *Connext* products, see the "3rdPartySoftware" documents in your installation: **<NDDSHOME>/doc/manuals/connext_dds_professional/release_notes_3rdparty**.