

RTI Connex

What's New in Version 7.2.0



Trademarks

RTI, Real-Time Innovations, Connex, Connex Drive, NDDS, the RTI logo, 1RTI and the phrase, “Your Systems. Working as one.” are registered trademarks, trademarks or service marks of Real-Time Innovations, Inc. All other trademarks belong to their respective owners.

Copy and Use Restrictions

No part of this publication may be reproduced, stored in a retrieval system, or transmitted in any form (including electronic, mechanical, photocopy, and facsimile) without the prior written permission of Real-Time Innovations, Inc. The software described in this document is furnished solely under and subject to RTI's standard terms and conditions available at <https://www.rti.com/terms> and in accordance with your License Acknowledgement Certificate (LAC) and Maintenance and Support Certificate (MSC), except to the extent otherwise accepted in writing by a corporate officer of RTI.

Third-Party Software

RTI software may contain independent, third-party software or code that are subject to third-party license terms and conditions, including open source license terms and conditions. Copies of applicable third-party licenses and notices are located at community.rti.com/documentation. IT IS YOUR RESPONSIBILITY TO ENSURE THAT YOUR USE OF THIRD-PARTY SOFTWARE COMPLIES WITH THE CORRESPONDING THIRD-PARTY LICENSE TERMS AND CONDITIONS.

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

1 What's New in 7.2.0

1.1 Connex Observability Framework Enhancements (Experimental)	3
1.1.1 Support for most observability backends with OpenTelemetry integration	3
1.1.2 Secure communications among Observability Framework components	3
1.1.3 Name change from "RTI Observability Library" to "RTI Monitoring Library 2.0"	3
1.1.4 Configure initial log forwarding verbosity via MONITORING QoS policy	3
1.1.5 Set Collector initial peers in MONITORING QoS policy	4
1.2 Instance State Consistency Enhancements	5
1.2.1 Instance state consistency with security fully supported	5
1.2.2 Integration with Admin Console's data visualization feature	5
1.2.3 Instance state consistency now more robust	6
1.2.4 Improvements in resource management while using instance state consistency	6
1.2.5 "Instance state recovery" feature and QoS name are now "instance state consistency"	7
1.3 Simple Participant Discovery Protocol 2.0 Enhancements	7
1.3.1 Improved partition change synchronization for enhanced discovery and liveness in SPDP2	7
1.3.2 Enhanced liveness communication with RTPS peer descriptors in SPDP2 improves liveness with Cloud Discovery Service	7
1.3.3 More easily enable SPDP2 using new QoS value in builtin_discovery_plugins	8
1.4 Python API Now Fully Productized and Supported	8
1.5 Other API Improvements	8
1.5.1 Request-Reply API for Python available	8
1.5.2 Modern C++ API	9
1.6 Support for systems running beyond 2038	9
1.7 Platform and Build Changes	9
1.7.1 Support for Limited Bandwidth Plugins and Limited Bandwidth Endpoint Discovery on certain Linux Platforms	9
1.7.2 Support for Security Plugins for wolfSSL 5.5.1 on certain Linux platforms	10

1.7.3 Support for Java 17	10
1.7.4 Support for Java API on macOS 11 and 12 on Arm v8 CPUs	10
1.7.5 Support for rtiddsgen_server on macOS	10
1.7.6 New components and imported target libraries for Connex Test Framework added to "FindPackage" CMake script	10
1.7.7 Future platform changes	11
1.8 Deprecations and Removals	13
1.8.1 RTI Connector for Python removed in this release	13
1.8.2 Durable Writer History and Durable Reader State properties deprecated in this release	14
1.8.3 OpenSSL 1.1.1 support removed in this release	14
1.8.4 Deprecated computed_crc_kind property	14
1.9 Product Availability Changes	14
1.9.1 RTI Database Integration Service not included in this release	14
1.10 Other	14
1.10.1 Convert between BuiltinTopicKey and InstanceHandle using new C and Traditional C++ API functions ..	14
1.10.2 Receive thread behavior is more deterministic and debuggable	15
1.10.3 Support for Distributed Logger in Modern C++	15
1.10.4 Durable Writer History and Durable Reader State can now be persisted in SQLite database	15
1.10.5 Support for RTPS Header Extension	16
1.10.6 Variables no longer created for each baseline QoS	16
2 Previous Releases	
2.1 What's New in 7.1.0	17
2.1.1 Monitor Health of Connex Applications Using New Connex Observability Framework (Experimental) ...	19
2.1.2 Enable Consistent View of Instance States Despite Disruptions to Connectivity, using New QoS Parameter (Experimental)	23
2.1.3 Simple Participant Discovery Protocol 2.0 Improvements	26
2.1.4 API Improvements	28
2.1.5 Debuggability Improvements	30
2.1.6 Platform and Build Changes	31
2.1.7 Deprecations and Removals	32
2.1.8 Product Availability Changes	33
2.1.9 Other	34
2.2 What's New in 7.0.0	35
2.2.1 Discovery Scalability and Troubleshooting	37
2.2.2 Language Bindings, APIs, XML Configuration	40
2.2.3 Usability	43
2.2.4 Scalability	46

2.2.5 Bandwidth Sensitivity	48
2.2.6 Security Improvements	50
2.2.7 Logging Improvements	51
2.2.8 DDS Spy and DDS Ping Improvements	56
2.2.9 Platform and Build Changes	59
2.2.10 Changes to Defaults	60
2.2.11 Performance Improvements	60
2.2.12 Deprecations and Removals	61
2.2.13 Product Availability Changes	64
2.2.14 Third-Party Software Upgrades	65




1 What's New in 7.2.0




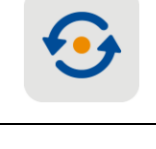

RTI® Connex® 7.2.0 is a feature release based on release 7.1.0. See the [Connex Releases](#) web page on the RTI website for more information. This document highlights new features, platforms, and improvements in *Connex*, including the Core Libraries, for 7.2.0.

For what's *fixed* in the Core Libraries for 7.2.0, see the [RTI Connex Core Libraries Release Notes](#).

For backward compatibility information between 7.2.0 and previous releases, see the *Migration Guide* on the RTI Community Portal (<https://community.rti.com/documentation>).

Table 1.1 Release 7.2.0 Highlights

	<p>Support for systems running beyond 2038</p> <p><i>Connex</i> applications now support systems beyond 2038, following the latest OMG DDS and RTPS specifications. This update enables <i>Connex</i> applications to run smoothly until 2106.</p>
	<p>Python API Now Fully Productized and Supported</p> <p>Python joins the ranks as a fully supported API along with C, Modern C++, Java, and .NET. This release also adds pip-based installation and support for Request-Reply.</p>
	<p>Connex Observability Framework Enhancements (Experimental)</p> <p><i>Observability Framework's</i> integration with OpenTelemetry enables flexible storage and analysis of telemetry data in a variety of third-party observability backends. Developers can leverage OpenTelemetry's vast ecosystem to store and analyze telemetry data in their preferred systems.</p>

	<p>Graphical Data Publishing in Admin Console</p> <p>Enhancements include altering QoS after publication, dragging and dropping samples between sample log and sample queue, importing and exporting samples, and more. See the RTI Admin Console Release Notes.</p>
	<p>Lightweight Security Enhancements</p> <p><i>Connex</i>t applications running the regular <i>Security Plugins</i> can now interoperate with Lightweight Security applications. Additionally, Lightweight Security can be applied to <i>Cloud Discovery Service</i> and <i>Real-Time WAN Transport</i>, ensuring secure data transmission in cloud environments. See the RTI Security Plugins Release Notes.</p>
	<p>Dynamic Certificates Renewal and Revocation Enhancements</p> <p>Dynamic Access Control can be created based on dynamic Identity Certificates, Certificate Revocation Lists (CRL), or a whitelist of Identity Certificate Subject Names. Furthermore, Dynamic Certificates are seamlessly integrated with RTI Infrastructure Services and <i>Admin Console</i>. See the RTI Security Plugins Release Notes.</p>
	<p>Instance State Consistency Enhancements</p> <p>Instance state consistency is now integrated with the <i>Security Plugins</i> and with <i>Admin Console</i>'s data visualization feature. It also has improved resource management and other features.</p>
	<p>Simple Participant Discovery Protocol 2.0 Enhancements</p> <p>The upgraded SPDP2 (Simple Participant Discovery Protocol 2.0) enhances robustness and seamless integration with the <i>Security Plugins</i>. It introduces support for IP mobility events, secure participant sample signature verification, reliable partition change handling, and compatibility with <i>Cloud Discovery Service</i>.</p>

For what's new and fixed in other products included in the *Connex*t suite, see those products' release notes on <https://community.rti.com/documentation> or in your installation. Or find those release notes here:

- [RTI Connex](#)t Core Libraries Release Notes
- [RTI Admin Console Release Notes](#)
- [RTI Code Generator Release Notes](#)
- [RTI Routing Service User's Manual](#)
- [RTI Security Plugins Release Notes](#)
- [RTI Observability Framework Release Notes](#)
- [RTI Cloud Discovery Release Notes](#)
- [RTI Real-Time WAN Transport Release Notes](#)
- [RTI Recording Service User's Manual](#)
- [RTI Persistence Service Release Notes](#)

- [RTI Web Integration Service Release Notes](#)
- [RTI Launcher Release Notes](#)
- [RTI Monitor Release Notes](#)
- [RTI Shapes Demo User's Manual](#)
- [RTI System Designer Getting Started Guide](#)
- [RTI Limited Bandwidth Plugins Release Notes](#)
- [RTI TLS Support Release Notes](#)

See also [1.9 Product Availability Changes on page 14](#).

1.1 Connex Observability Framework Enhancements (Experimental)

This release adds the following new capabilities to *RTI Connex Observability Framework*, which was introduced in release 7.1.0 (see [2.1.1 Monitor Health of Connex Applications Using New Connex Observability Framework \(Experimental\)](#) on page 19).

See also the *RTI Observability Framework* documentation.

1.1.1 Support for most observability backends with OpenTelemetry integration

For details, see the [RTI Observability Framework Release Notes](#).

1.1.2 Secure communications among Observability Framework components

For details, see the "Security" section in the Getting Started Guide chapter of the *RTI Observability Framework* documentation and the [RTI Observability Framework Release Notes](#).

1.1.3 Name change from "RTI Observability Library" to "RTI Monitoring Library 2.0"

This release changes the name of the "RTI Observability Library" to "RTI Monitoring Library 2.0."

This change includes changing the names of builtin QoS profiles. If your application references these profiles, you should update the names accordingly. See the *Migration Guide* on the RTI Community Portal (<https://community.rti.com/documentation>) for details.

1.1.4 Configure initial log forwarding verbosity via MONITORING QoS policy

The forwarding verbosity controls the level of log messages an application forwards to *RTI Observability Collector Service*, for use by *Observability Framework*.

In the previous release, this verbosity level could only be changed through the Dashboard, with "warnings" as the default value. In this release, applications' initial forwarding verbosity level can be

configured using a new field in the MONITORING QoS policy: `monitoring::telemetry_data::logs::middleware_forwarding_level` (the rest of the Syslog Facilities are not supported yet). This way, you can forward log messages with the desired verbosity right away from the beginning, rather than using the Dashboard to change the verbosity later.

This verbosity level can be specified in code or through XML (`USER_QOS_PROFILES.xml`):

```
<participant_factory_qos>
  <monitoring>
    ....
    <telemetry_data>
      <logs>
        <middleware_forwarding_level>NOTICE</middleware_forwarding_level>
      </logs>
    </telemetry_data>
  </monitoring>
</participant_factory_qos>
```

For more information on the `middleware_forwarding_level` field, see the *MONITORING QoS Policy* section of the [RTI Connex Core Libraries User's Manual](#).

1.1.5 Set Collector initial peers in MONITORING QoS policy

RTI Monitoring Library 2.0 (previously known as the RTI Observability Library) creates a dedicated *DomainParticipant* in the Observability Domain ID to distribute telemetry data for use by *Observability Framework*.

Setting the initial peers list for this *DomainParticipant* wasn't trivial. You had to create an additional QoS profile inheriting from `BuiltinQosLib::Generic.Observability`, set the initial peers list in that new profile, and refer to that profile using the `distribution_settings.dedicated_participant.participant_qos_profile_name` field in the MONITORING QoS policy.

Since setting the initial peers pointing to the *RTI Observability Collector Service* is always required, in this release *Connex* now has a new field, `collector_initial_peers`, under `distribution_settings.dedicated_participant`, that makes the configuration easier.

Here is an example that shows how to set the field in an XML configuration file:

```
<qos_profile name="MyApplicationProfile">
  <participant_factory_qos>
    <monitoring>
      <enable>true</enable>
      <distribution_settings>
        <dedicated_participant>
          <collector_initial_peers>
            <element>192.168.1.1</element>
          </collector_initial_peers>
        </dedicated_participant>
      </distribution_settings>
    </monitoring>
```

```
</participant_factory_qos>
</qos_profile>
```

For more information on the **collector_initial_peers** field, see the *MONITORING QoS Policy* section of the [RTI Connext Core Libraries User's Manual](#).

1.2 Instance State Consistency Enhancements

This release improves the new capability introduced in [2.1.2 Enable Consistent View of Instance States Despite Disruptions to Connectivity, using New QoS Parameter \(Experimental\)](#) on page 23.

Instance state consistency is no longer experimental starting in release 7.2.0.

For details on instance state consistency, see the "Transition after NOT_ALIVE_NO_WRITERS" section in *Instance States*, in the [RTI Connext Core Libraries User's Manual](#).

1.2.1 Instance state consistency with security fully supported

Security can now be enabled in systems that enable instance state consistency.

A new domain-level rule, **instance_state_consistency_protection_kind**, has been added to the governance file in the *Security Plugins* that allows the protection kind of the instance state consistency channel to be configured. If **instance_state_consistency_protection_kind** is not set in the governance file, the protection kind is inherited from **discovery_protection_kind**. When the protection kind is inherited from discovery, the **instance_state_consistency_protection_kind** is not propagated during discovery.

Note that the **instance_state_consistency_protection_kind** only configures the submessage protection used by the channel. The payload protection uses the crypto data from the *DataWriter* with which the *DataReader* has regained liveliness. This is to ensure that the topic-level security configured in a system is respected.

To enable security with instance state consistency, the service request channel must also have security enabled (since the service request channel is used to send the request for instance state data).

For more information about **instance_state_consistency_protection_kind**, see "Domain-Level Rules," in the *RTI Security Plugins User's Manual*.

1.2.2 Integration with Admin Console's data visualization feature

Admin Console's Instance Table will now attempt to recover and display samples that have transitioned back to the ALIVE state. Note that because only a single sample is kept for each instance, this recovery is not always possible. For example, if a different *DataWriter* with a lower strength publishes samples after the liveliness loss, then *Admin Console's* cached sample will be from the lower-strength *DataWriter* and so will not show in the Instance Table. However, if the cached sample is from the same *DataWriter* that is transitioning the instance to ALIVE, then the data will appear.

1.2.3 Instance state consistency now more robust

The following changes enhance robustness of the instance state consistency feature.

1.2.3.1 Instance key values now always available for dispose and "valid data" samples

You can now always call `get_key_value` to determine which instance has transitioned when a sample with `valid_data=FALSE` is received, as long as the instance has been seen by the *DataReader* before.

Previous to this change, if the instance had previously been detached, then a call to `get_key_value` would have failed to retrieve the key value. In the context of instance state consistency, this meant that when a writer of an instance regained liveliness after a network disconnection, and the instance transitioned back to ALIVE with `valid_data=FALSE`, it was not possible to call `get_key_value` to identify the instance that was transitioning back to ALIVE. Now, the key value can be retrieved in this situation as long as `keep_minimum_state_for_instances=TRUE` in the *DataReader's* `DATA_READER_RESOURCE_LIMITS` QoS policy.

1.2.3.2 Instance state consistency correctly handles instances removed from *DataWriter* queue during disconnection

In release 7.1.0, instances that were removed from the *DataWriter* queue during a disconnection may have transitioned to their last known state on the *DataReader*, resulting in inconsistent behavior. Now, instances that are removed from the *DataWriter* queue during a disconnection no longer transition to any state after a reconnection. This is because, during the disconnection, the *DataReader* may have missed an instance transition for the removed instance, and therefore the *DataReader* can't know which state to transition it to. Furthermore, the *DataWriter* can't tell the *DataReader* what the final state was because it has removed all information about that instance from its queue.

See the "Transition after NOT_ALIVE_NO_WRITERS" section in *Instance States*, in the [RTI Connext Core Libraries User's Manual](#) for a complete explanation of this and other special considerations.

1.2.4 Improvements in resource management while using instance state consistency

The following enhancements have been made to improve resource management related to instance state consistency:

- A new QoS setting, `autopurge_remote_not_alive_writer_delay`, prevents unbounded memory growth related to instance state consistency. See the *DATA_READER_RESOURCE_LIMITS QosPolicy (DDS Extension)* section of the [RTI Connext Core Libraries User's Manual](#) for more details.
- This release prevents unbounded memory growth on the *DataWriter*, which occurred when using instance state consistency in the previous release.

1.2.5 "Instance state recovery" feature and QoS name are now "instance state consistency"

The name of the "instance state recovery" feature has been changed to "instance state consistency" to more accurately reflect its benefit. This change includes renaming some fields (e.g., in the C API, **instance_state_recovery_kind** in the RELIABILITY QoS policy changed its name to **instance_state_consistency_kind**) and some values (e.g., `DDS_RECOVER_INSTANCE_STATE_RECOVERY` became `DDS_RECOVER_INSTANCE_STATE_CONSISTENCY`, and `DDS_NO_INSTANCE_STATE_RECOVERY` became `DDS_NO_RECOVER_INSTANCE_STATE_CONSISTENCY`).

1.3 Simple Participant Discovery Protocol 2.0 Enhancements

The following improvements have been made to the new feature described in [2.1.3 Simple Participant Discovery Protocol 2.0 Improvements on page 26](#).

For details on Simple Participant Discovery Protocol 2.0 (SPDP2), see the *Simple Participant Discovery 2.0* section of the [RTI Connext Core Libraries User's Manual](#).

SPDP2 is no longer experimental starting in release 7.2.0.

1.3.1 Improved partition change synchronization for enhanced discovery and liveliness in SPDP2

In previous releases of SPDP2, a partition change between two matching *DomainParticipants* that led to the *DomainParticipants*' not matching was sent reliably on the *ParticipantConfigBuiltinTopicData Topic*. However, the local *DomainParticipant* changing its partitions did not wait for the remote *DomainParticipant* to receive the partition change. If the local *DomainParticipant* configuration message containing the partition change was lost and the remote *DomainParticipant* discovery locator was not part of the local *DomainParticipant*'s initial peers, the remote *DomainParticipant* would not discover the partition change until it lost liveliness with the local *DomainParticipant*.

This behavior has changed in release 7.2.0. Now, when using SPDP2, the local *DomainParticipant* will wait until the remote *DomainParticipant* acknowledges reception of the configuration change before fully removing the remote participant.

1.3.2 Enhanced liveliness communication with RTPS peer descriptors in SPDP2 improves liveliness with Cloud Discovery Service

DomainParticipants using Simple Participant Discovery Protocol 2.0 (SPDP2) were previously at risk of losing liveliness with any Infrastructure Service that was configured to communicate with an RTPS peer descriptor.

DomainParticipants use an RTPS peer descriptor in their initial peers when the participant needs to communicate with a service, but the service does not perform discovery with the participant. Participants now use RTPS peer descriptors to communicate with *Cloud Discovery Service* (CDS). In this

case, a participant sends its participant announcements to CDS and CDS forwards those announcements to the other participants, but CDS will never send a participant announcement about itself and therefore will never perform participant discovery with the participant. Because participant discovery does not take place, the participant never sends liveliness messages to CDS.

Since the participant was previously only sending participant announcements at the **participant_announcement_period**, CDS may have lost liveliness with the participant if the **participant_announcement_period** was greater than the **participant_liveliness_lease_duration**.

A participant using SPDP2 will now send additional bootstrap messages to any RTPS peer descriptors in its initial peers at the **participant_liveliness_assert_period** if the **participant_liveliness_assert_period** is less than the **participant_announcement_period**. This allows the participant to maintain liveliness with any service by sending out updates at a period that is less than the **participant_liveliness_lease_duration**.

1.3.3 More easily enable SPDP2 using new QoS value in builtin_discovery_plugins

Previously, to enable SPDP2, you had to provide **SPDP2 | SEDP** as the **builtin_discovery_plugins** mask in the DISCOVERY_CONFIG QoS Policy. (SPDP2 must be enabled with an endpoint discovery plugin, such as the Simple Endpoint Discovery Protocol (SEDP).) A new value, **SDP2**, has been added to the valid inputs for this field to enable both SPDP2 and SEDP. This is consistent with the existing **SDP** value, which enables both SPDP and SEDP.

1.4 Python API Now Fully Productized and Supported

The *Connex* Python API is no longer experimental in this release. With release 7.2.0, it is now a fully supported language API and can be used in production systems.

Starting with this release, the Python API is installed with pip.

To get started with the Python API, see the Python language version of the [RTI Connex Getting Started Guide](#). For further details, see the [Python API Reference HTML documentation](#).

1.5 Other API Improvements

1.5.1 Request-Reply API for Python available

This release includes the Request-Reply API for Python. For more information see [Request-Reply, in the Python API Reference HTML documentation](#).

1.5.2 Modern C++ API

1.5.2.1 Support for sequences of wchars in DynamicData

A sequence of wchars can now be set using `DynamicData::set_values`. Previously, only uint16s could be in an array, but now there is the flexibility to use both.

1.5.2.2 Exception messages now contain the name of the exception type

Exceptions thrown by the Modern C++ API now contain the name of the exception in the `what()` description.

1.6 Support for systems running beyond 2038

*Connex*t applications have added support for systems running beyond the year 2038. This update, which is compliant with the latest [OMG Real-Time Publish-Subscribe \(RTPS\) specification, version 2.5](#) (and was introduced in version 2.3), now makes it possible to run *Connex*t applications up to year 2106. This update applies only to this and future releases of *Connex*t; there are currently no plans to backport it to previous releases.

1.7 Platform and Build Changes

This release adds support for the following platforms, compared to release 7.1.0.

To see which products support the new platforms, see the Supported Platforms section of the [RTI Connex Core Libraries Release Notes](#).

Table 1.2 New Platforms

OS	OS Version	CPU	Toolchain	RTI Architecture
VxWorks®	VxWorks 22.09	x64	llvm 13.0.1.3	x64Vx22.09llvm13.0.1.3 x64Vx22.09llvm13.0.1.3_rtp
Linux®	Ubuntu® 18.04 LTS	Arm® v7	gcc 7.5.0	armv7Linux4gcc7.5.0

Note: Ubuntu 18.04 LTS for Arm v7 is supported in release 6.1.1/6.1.2. It was not in release 7.0.0/7.1.0; now, it is added back in 7.2.0.

1.7.1 Support for Limited Bandwidth Plugins and Limited Bandwidth Endpoint Discovery on certain Linux Platforms

This release add supports for *Limited Bandwidth Plugins* and *Limited Bandwidth Endpoint Discovery* to these platforms when using Arm v8 CPUs:

- Ubuntu 18.04 LTS
- Ubuntu 22.04 LTS

The RTI architecture for these platforms is armv8Linux4gcc7.3.0.

1.7.2 Support for Security Plugins for wolfSSL 5.5.1 on certain Linux platforms

Previously, the *Security Plugins* for wolfSSL were only supported in the armv8QNX7.1qcc_gpp8.3.0 target architecture. This release of the *Security Plugins* introduces support for wolfSSL® 5.5.1 on these platforms when using x64 CPUs:

- Red Hat® Enterprise Linux 8.0 and 9.0
- Ubuntu 18.04 LTS, 20.04 LTS, and 22.04 LTS

The RTI architecture for these platforms is x64Linux4gcc7.3.0.

Please see the instructions in the [RTI Security Plugins Installation Guide](#) to get started with the *Security Plugins* for wolfSSL.

1.7.3 Support for Java 17

Starting in this release, RTI's Java® libraries are compiled with AdoptOpenJDK® 17, with backward compatibility with Java 8. The JRE used by RTI Tools (such as *Admin Console*) is also upgraded to the JRE from OpenJDK™ 17.

1.7.4 Support for Java API on macOS 11 and 12 on Arm v8 CPUs

The Java API is now supported when using macOS 11 and 12 architectures on Arm® V8 CPUs (target architecture arm64Darwin20clang12.0).

1.7.5 Support for rtiddsgen_server on macOS

RTI Code Generator's **rtiddsgen_server** is now supported on macOS. Previously, it was supported only on Windows and Linux. This support was added in release 7.1.0, but not documented at that time.

1.7.6 New components and imported target libraries for Connex Test Framework added to "FindPackage" CMake script

Connex Test Framework libraries are now installed with the target libraries of each architecture. For details on the *Connex* Test Framework, which is for use with the *Security Plugins SDK*, see the [RTI Security Plugins Release Notes](#).

To be able to link against the two new imported targets, two new components have been added to the "Find Package" script (**FindRTIConnexDDS.cmake**):

- **test** component:
 - **RTIConnex DDS::test** containing the librtitest library
- **test_helpers** component:
 - **RTIConnex DDS::test_helpers** containing **libnddsctesthelpers** and **libnddsctrans-porttesthelpers** libraries

1.7.7 Future platform changes

This section describes:

- [Standard platforms planned for Connex 7.3.0 LTS](#)
- [Libraries transitioning from standard to custom support](#)

1.7.7.1 Standard platforms planned for Connex 7.3.0 LTS

The following table shows the provisional list of standard libraries that will be supported in *Connex* 7.3.0 LTS when it is released in 2024. Please note that this table is subject to change; the final version will be published in the *Release Notes* for *Connex* 7.3.0 LTS.

Table 1.3 Future Supported Standard Platforms

OS	OS Version	CPU	Toolchain	RTI Architecture
Android™	Android 12	Arm v8	clang 12.0.8 (ndk r23b)	arm64Android12clang12.0.8ndkr23b
	Android 14	Arm v8	clang (version TBD)	TBD
Linux	Red Hat® Enterprise Linux 8, 9 Ubuntu 18.04 LTS,, 20.04 LTS, 22.04 LTS	x64	gcc 7.3.0	x64Linux4gcc7.3.0
	Ubuntu 22.04 LTS	x64	clang 15.0.1	x64Linux5clang15.0.1
	Ubuntu 18.04 LTS	Arm v7	gcc 7.5.0	armv7Linux4gcc7.5.0
	Ubuntu 18.04 LTS, 22.04 LTS	Arm v8	gcc 7.3.0	armv8Linux4gcc7.3.0
macOS®	macOS 11, 12, 13	x64	clang 12.0, 13.0, 14.0	x64Darwin20clang12.0
		Arm v8	clang 12.0, 13.0, 14.0	arm64Darwin20clang12.0

OS	OS Version	CPU	Toolchain	RTI Architecture
QNX®	QNX Neutrino® 7.1	x64	qcc_cxx 8.3.0 (LLVM C++ library)	x64QNX7.1qcc_cxx8.3.0
		Arm v8	qcc_gpp 8.3.0 (GNU C++ library)	armv8QNX7.1qcc_gpp8.3.0
	QNX Neutrino 8.0	x64	qcc_cxx (version TBD) (LLVM C++ library)	TBD
		Arm v8	qcc_gpp (version TBD) (GNU C++ library)	TBD
VxWorks®	VxWorks 23.09	x64	llvm (version TBD)	TBD
				TBD (_rtp)
Windows®	Windows 10, 11, Windows Server 2016, 2022	x64	VS 2017, VS 2019, VS 2022	x64Win64VS2017
	Windows 11	Arm v8	VS 2022	TBD

RTI welcomes your feedback regarding new platforms that you may need for your next projects. Please contact us at platforms-pm@rti.com.

1.7.7.2 Libraries transitioning from standard to custom support

In *Connex* 7.3.0 LTS, the following libraries will no longer appear as standard libraries. However, they will not disappear entirely; we are aware that they may still be necessary for specific projects. Therefore we will continue to offer them as Custom Target Libraries (CTLs) for a limited time.

[Table 1.4 Libraries Transitioning from Standard to Custom Support](#) below lists libraries that were standard in *Connex* 6.1.2 LTS, but will be custom libraries in *Connex* 7.3.0 LTS. Other CTLs may also be available; contact your sales representative for details. If you are interested in using one of these platforms, please contact your local RTI sales representative or email sales@rti.com.

Table 1.4 Libraries Transitioning from Standard to Custom Support

OS	OS Version	CPU	Toolchain	RTI Architecture
Linux	Red Hat Enterprise Linux 7, 7.3, 7.5, 7.6 CentOS 7.0	x64	gcc 4.8.2	x64Linux3gcc4.8.2
	Red Hat Enterprise Linux 7, 7.3, 7.5, 7.6 CentOS 7.0	x86	gcc 4.8.2	i86Linux3gcc4.8.2

OS	OS Version	CPU	Toolchain	RTI Architecture
QNX	QNX Neutrino 7.1	Arm v8	qcc_cxx 8.3.0	armv8QNX7.1qcc_cxx8.3.0
	QNX Neutrino 7.0.4	x64	qcc_gpp 5.4.0	x64QNX7.0.0qcc_gpp5.4.0
		Arm v8	qcc_cxx 5.4.0	armv8QNX7.0.0qcc_cxx5.4.0
VxWorks	VxWorks 7.0 (SR0630)	x64	llvm 8.0.0.2	x64Vx7SR0630llvm8.0.0.2
				x64Vx7SR0630llvm8.0.0.2_rtp
Windows	Windows 10 Windows Server 2012 R2, 2016	x64	VS 2015	x64Win64VS2015
	Windows 10 Windows Server 2016	x86	VS 2017 VS 2019	i86Win32VS2017
			VS 2015	i86Win32VS2015

If you have any questions regarding the differences between a Target Library Standardization (TLS) and a CTL, please consult your sales representative.

1.8 Deprecations and Removals

This section describes products, features, and platforms that are *deprecated* or *removed* starting in release 7.2.0.

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.

1.8.1 RTI Connector for Python removed in this release

Now that the Python API is fully supported, *Connector for Python* is removed in release 7.2.0. (It was deprecated in 7.0.0.) See [1.4 Python API Now Fully Productized and Supported on page 8](#). *Connector for Javascript* is still supported.

1.8.2 Durable Writer History and Durable Reader State properties deprecated in this release

Because of the enhancements described in [1.10.4 Durable Writer History and Durable Reader State can now be persisted in SQLite database on the next page](#), this release deprecates the **dds.data_writer.history.*** and **dds.data_reader.state.*** properties (except for the **dds.data_reader.state.persistence_service.request_depth** property). Although these deprecated properties are still available for use, you should use the fields in the DURABILITY QoS policy instead to configure the durable writer history and durable reader state. The deprecated properties may be removed in a future release.

1.8.3 OpenSSL 1.1.1 support removed in this release

The support of OpenSSL 1.1.1 has been removed, because it is end-of-life in September, 2023. See also the [RTI Security Plugins Release Notes](#), [RTI TLS Support Release Notes](#), and [RTI Web Integration Service Release Notes](#).

1.8.4 Deprecated computed_crc_kind property

The CRC_32_LEGACY option value, used by the **computed_crc_kind** property, is deprecated starting in release 7.2.0. See [1.10.5 Support for RTPS Header Extension on page 16](#) for more information.

1.9 Product Availability Changes

1.9.1 RTI Database Integration Service not included in this release

Database Integration Service is not included in release 7.2.0. You can use *Database Integration Service* versions 6.1.2 or older. If you use those versions with a *Connex7* release, see the *Migration Guide* on the RTI Community Portal (<https://community.rti.com/documentation>); look for the "Wire Protocol" sections to find wire protocol compatibility issues, if any, between the older release and the *Connex7* release. Address these issues before using *Database Integration Service* with *Connex7*.

1.10 Other

1.10.1 Convert between BuiltinTopicKey and InstanceHandle using new C and Traditional C++ API functions

When using the C or Traditional C++ API, you may sometimes want to convert a **BuiltinTopicKey** to an **InstanceHandle**. For example, suppose you call **DDS_DataWriter_get_matched_subscription_participant_data** to get the **DDS_BuiltinTopicKey_t** of the *DomainParticipant* whose *DataReader* is matched with your *DataWriter*, and then you want to call **DDS_DomainParticipant_ignore_participant** on that *DomainParticipant*. This release introduces the API function **DDS_BuiltinTopicKey_to_instance_handle**, which allows you to get the **InstanceHandle** that you can pass into a function such as **DDS_DomainParticipant_ignore_participant**. For completeness, this release also introduces the API function **DDS_BuiltinTopicKey_from_instance_handle** to do the opposite conversion.

1.10.2 Receive thread behavior is more deterministic and debuggable

*Connex*t relies on a thread pool to service transport receive resources. In previous releases, the threads of this pool were not bound to any specific resource, and therefore any thread could serve any transport receive resource at any time.

This release changes the way the receive threads are managed: receiver threads are now bound to specific transport receive resources. *Connex*t creates a specific thread to serve each resource, and the lifecycle of the thread is tied to the lifecycle of the associated receive resource. As a result of this change, receive thread behavior is more deterministic and debuggable, and it is possible for *Connex*t applications to (subject to the operating system's capabilities) change the thread configuration for specific receive resources.

1.10.3 Support for Distributed Logger in Modern C++

Previously, Distributed Logger was only available in C, Traditional C++, and Java. Distributed Logger functionality is now fully supported by the Modern C++ API. The new Distributed Logger API has also been redesigned to be used more intuitively and to follow Modern C++ best-practices. See the [RTI Distributed Logger Modern C++ API Reference](#).

1.10.4 Durable Writer History and Durable Reader State can now be persisted in SQLite database

Before *Connex*t 6.1.1, the ability to persist the historical cache of a *DataWriter* and the state of a *DataReader* was only supported with external relational databases such as MySQL. In 6.1.1 the support for external databases was deprecated, and support for Durable Writer History and Durable Reader State was temporarily disabled. This release reintroduces these two features by providing file-based storage using an SQLite database.

In addition to allowing persisting the Durable Writer History and Durable Reader State into SQLite files, this release simplifies the configuration of these features by going away from property-based configuration and introducing a new QoS field in the DURABILITY QoS policy, **storage_settings**. This field can be used for *DataWriters* and *DataReaders*. It can be configured programmatically or via XML like any other QoS value. For example, enabling Durable Writer History is as simple as this:

```
<datawriter_qos>
  <durability>
    <persistent_storage>
      <enable>true</enable>
      <file_name>MyDB.db</file_name>
    </persistent_storage>
  </durability>
</datawriter_qos>
```

For more information, see the *Mechanisms for Achieving Information Durability and Persistence* chapter of the [RTI Connex Core Libraries User's Manual](#).

1.10.5 Support for RTPS Header Extension

This release adds support for the RTPS Header Extension, defined in the [OMG Real-Time Publish-Subscribe \(RTPS\) specification, version 2.5](#). The Header Extension is a special submessage that can be added to every RTPS message. It contains metadata about the entire RTPS message, which can be useful for diagnostics and debugging. Introduction of this feature deprecates the rti-specific RTPS CRC and makes the new, RTPS-specification-compliant CRC32C, transported on the RTPS Header Extension, the new default. For details, see information about the **compute_crc** and **check_crc** fields in the *WIRE_PROTOCOL QosPolicy (DDS Extension)* section of the [RTI Connext Core Libraries User's Manual](#).

1.10.6 Variables no longer created for each baseline QoS

Prior to this release, variables were created for each new baseline QoS: for example, `DDS_PROFILE_BASELINE_7_1_0` for the 7.1.0 release. For this and future releases, these variables will no longer be created due to their limited usefulness.

2 Previous Releases




2.1 What's New in 7.1.0






RTI® Connex® 7.1.0 is a feature release based on release 7.0.0. See the [Connex Releases](#) web page on the RTI website for more information. This document highlights new features, platforms, and improvements in *Connex*, including the Core Libraries, for 7.1.0.

For what's *fixed* in the Core Libraries for 7.1.0, see the [RTI Connex Core Libraries Release Notes](#).

For backward compatibility information between 7.1.0 and previous releases, see the *Migration Guide* on the RTI Community Portal (<https://community.rti.com/documentation>).

Table 2.1 Release 7.1.0 Highlights

	<p>Monitor Health of Connex Applications Using New Connex Observability Framework (Experimental)</p> <p>Provides a holistic solution for collecting telemetry data (metrics and logs) scalably, delivering the data to a host where it can be aggregated and visualized.</p>
	<p>Graphical Data Publishing (Experimental in 7.1.0)</p> <p>Allows writing data to a user <i>Topic</i>, using a lightweight language binding (Python) to update the fields. See the RTI Admin Console Release Notes.</p>
	<p>Lightweight Security</p> <p>Provides an option for pragmatic and minimal security enforcement: message authentication and encryption to address DDS specification vulnerability (where initial packets are not protected) and an improved security model for domain bootstrapping. See the RTI Security Plugins Release Notes.</p>

	<p>Support for Dynamic Certificate Expiration</p> <p>When a certificate expires, certificate owners will be automatically removed, enabling long-running, uninterrupted operation of <i>Connex</i> secure systems. See the RTI Security Plugins Release Notes.</p>
	<p>RTI Routing Service in RTI System Designer</p> <p><i>RTI System Designer</i> has new views for creating a <i>Routing Service</i> configuration in <i>System Designer</i>. See the RTI System Designer Getting Started Guide.</p>
	<p>Enable Consistent View of Instance States Despite Disruptions to Connectivity, using New QoS Parameter (Experimental)</p> <p>When an application regains liveliness after a disruption (such as connectivity loss), the state of all the instances subscribed to that application transition back to their state before the disconnection.</p>
	<p>Support for OpenSSL 3.0</p> <p>See the RTI Security Plugins Release Notes.</p>
	<p>Simple Participant Discovery Protocol 2.0 Improvements</p> <p>Enhancements have been made to the SPDP 2.0 option, which improves scalability with large numbers of <i>DomainParticipants</i> by sending optimized match and liveliness messages.</p>

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

- [RTI Connex Core Libraries Release Notes](#)
- [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 Cloud Discovery Release Notes](#)

See also [2.1.8 Product Availability Changes on page 33](#).

2.1.1 Monitor Health of Connex Applications Using New Connex Observability Framework (Experimental)

The *Connex* Observability Framework is a holistic solution that provides deep visibility into the current and past states of *Connex* applications, using telemetry data such as logs and metrics. This visibility makes it easier to proactively identify and resolve potential issues, providing a higher level of confidence in the reliable operation of the system.

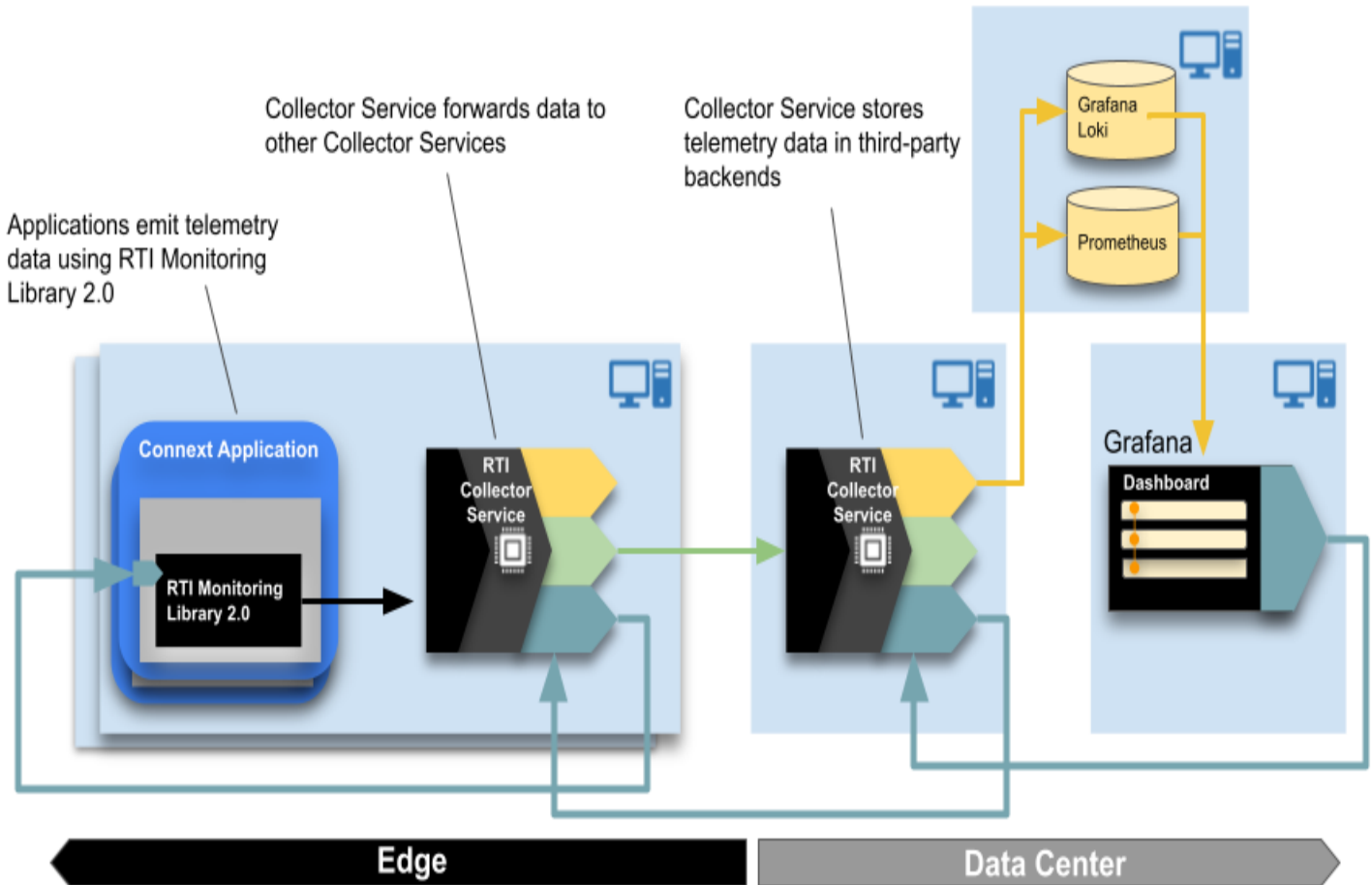
The *Connex* Observability Framework consists of three key components:

- RTI Monitoring Library 2.0 (formerly called RTI Observability Library) allows instrumenting a *Connex* application to emit telemetry data. The library also accepts remote commands (via the dashboards) to change the set of emitted telemetry data at runtime.
- RTI Observability Collector Service scalably collects and distributes telemetry data from individual *Connex* applications all the way to third-party telemetry backends, such as Prometheus® for metrics and Grafana® Loki™ for logs.
- RTI Observability Dashboards provide a way to visualize the telemetry data collected from *Connex* applications using a set of reference Grafana dashboards that you can customize or use as an example to enhance and build dashboards in the platform of your choice.

In this release, RTI supports Prometheus as the time-series database to store *Connex* metrics and Grafana Loki as the log aggregation system to store *Connex* logs. As OpenTelemetry is gathering widespread support in the observability industry, in future releases the Observability Framework will support OpenTelemetry to help integrate with various observability backends.

Note: All product components in the Observability Framework are experimental, so do not deploy them in production.

Figure 2.1: Observability Framework Architecture



Observability Framework use cases include:

- **Debugging:** find the cause of an undesired behavior or determine if the system meets performance needs during development.
- **CI/CD Monitoring:** assess the performance impact of code or configuration changes.
- **Monitoring-deployed applications:** confirm that your systems are running as expected and fix potential performance issues proactively.

RTI Connex Observability Framework must be downloaded and installed separately. Check the [RTI Customer portal](#) or contact support@rti.com for information on how to obtain an Observability Framework package.

For information on installing the Observability Framework, as well as a step-by-step guide for running it with the included example, see the *RTI Observability Framework* documentation.

2.1.1.1 Monitoring Library 2.0

RTI Monitoring Library 2.0 allows instrumenting a *Connex* application to emit telemetry data (logs and metrics).

Note: This library was previously called Observability Library; the name changed to RTI Monitoring Library 2.0 starting in *Connex* 7.2.0.

RTI Monitoring Library 2.0 includes the following key features:

- Collection and emission of *Connex* metrics and logs. Secure logs will be supported in future releases.
- Configuration using a new MONITORING QoS Policy. The QoS policy can be set programmatically or via XML.
- Runtime changes to the set of emitted telemetry data using remote commands coming from the RTI Observability Collector. In this release, the library only allows changing the emission and generation log verbosity level for a *Connex* application.
- Ability to enable and disable use of the RTI Monitoring Library 2.0 at runtime by changing the MONITORING QoS Policy.
- Lower overhead compared to the RTI Monitoring Library (described in the *RTI Monitoring Library* section of the [RTI Connex Core Libraries User's Manual](#)).

To learn more, see the *RTI Monitoring Library 2.0* section of the [RTI Connex Core Libraries User's Manual](#). See also the *RTI Observability Framework* documentation.

2.1.1.2 Observability Collector Service

The RTI Observability Collector Service scalably collects telemetry data emitted by RTI Monitoring Library 2.0 in a *Connex* application. The service can work in two modes:

- **Forwarder:** In this mode, the Collector forwards the telemetry data from a *Connex* application to other Collector instances. This mode is not supported in this release.
- **Storage:** In this mode, Collector stores the telemetry data in a third-party observability backend. In this release, the observability backend is Prometheus for metrics and Grafana Loki for logs. Future releases will allow integration with other third-party backends by using OpenTelemetry.

The Observability Collector Service includes the following key features:

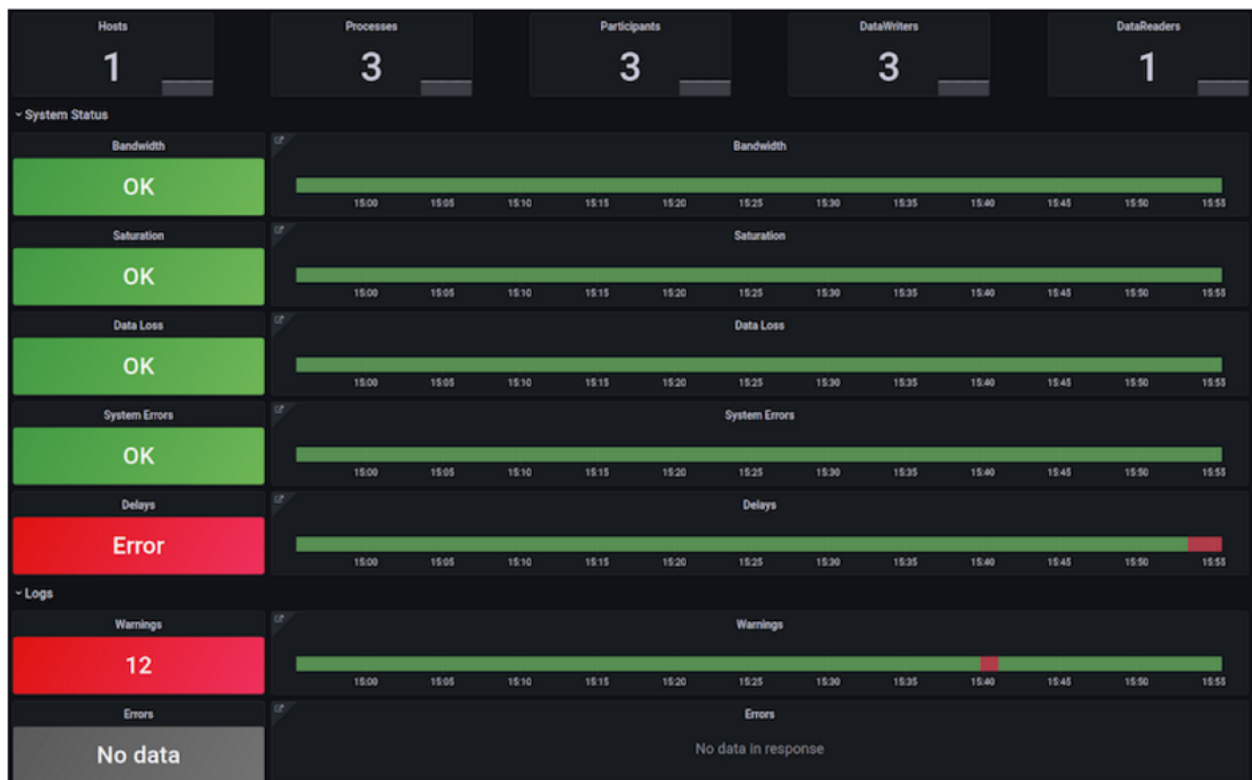
- Collection and filtering of telemetry data emitted by *Connex* applications (using RTI Monitoring Library 2.0) or other Collectors. This release does not provide filtering capabilities.
- Storage of telemetry data in third-party backends: Prometheus for metrics and Grafana Loki for logs.
- Remote command forwarding from the Observability dashboards to the *Connex* applications and resources to which the commands are directed. This release only allows forwarding commands that change the logging verbosity of *Connex* applications.

2.1.1.3 Observability Dashboards

This release includes a set of hierarchical Grafana dashboards that are built to alert you when a problem occurs and facilitate root cause analysis. The dashboards get the telemetry data that they visualize from the third-party backends in which RTI Collector Service stores the data.

The first layer of the Grafana dashboards provides a health status summary by focusing on five golden signals: Bandwidth, Saturation, Data Loss, System Errors, and Delays.

Figure 2.2: Grafana Dashboard with Errors



Starting on the first dashboard, you can get additional details on the error conditions by clicking on any of the golden signals showing a problem.

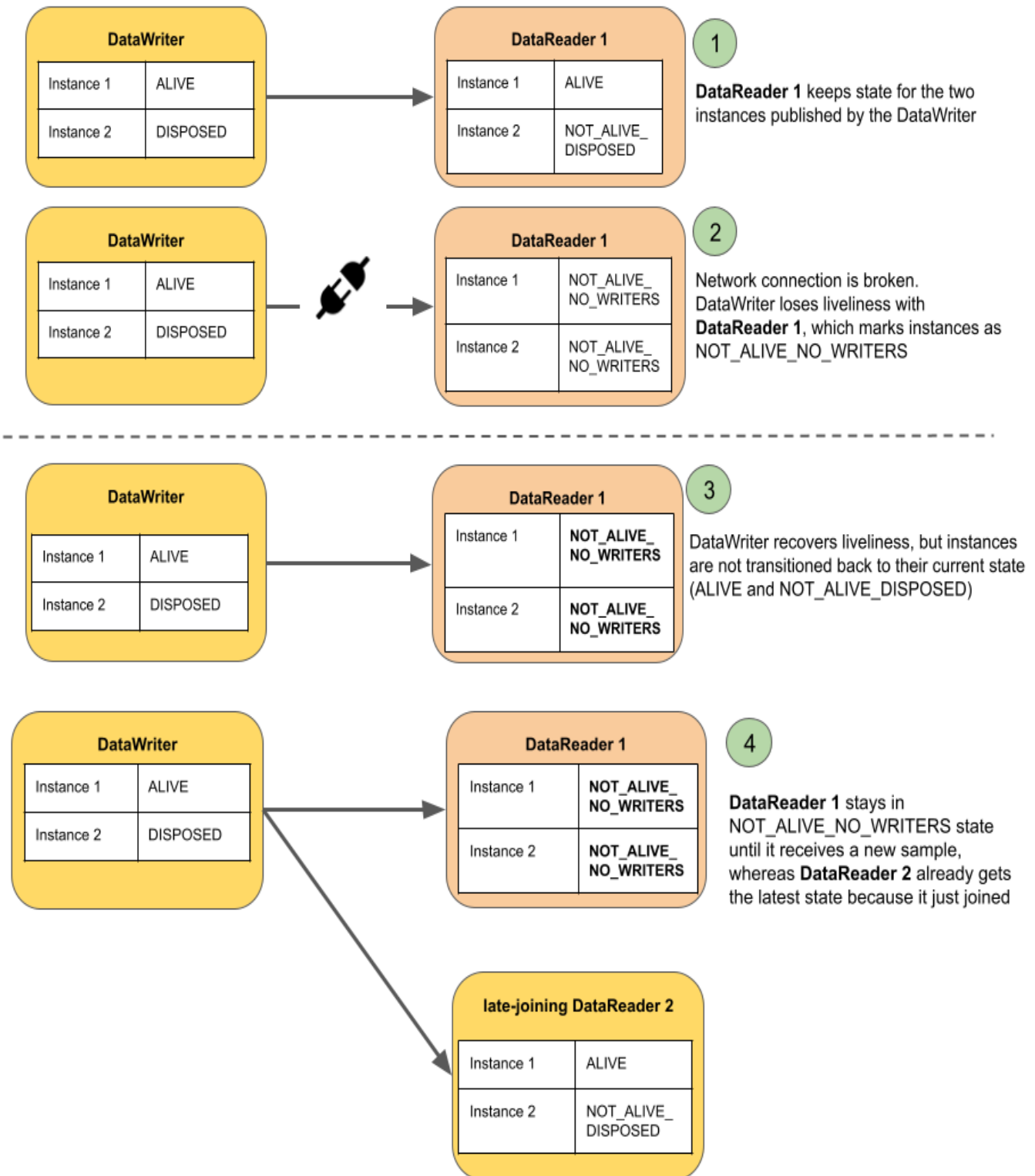
The top-level dashboard also provides access to the system logs and provides information about the number of entities running in the system.

2.1.2 Enable Consistent View of Instance States Despite Disruptions to Connectivity, using New QoS Parameter (Experimental)

The `NOT_ALIVE_NO_WRITERS` state for an instance indicates that there are no active *DataWriters* that are currently updating an instance. This can occur because all of the *DataWriters* that were publishing the instance have unregistered themselves from the instance or they all have become not alive (through losing liveness or being deleted). In this case, the *DataReader* will set the state of the instance to `NOT_ALIVE_NO_WRITERS`.

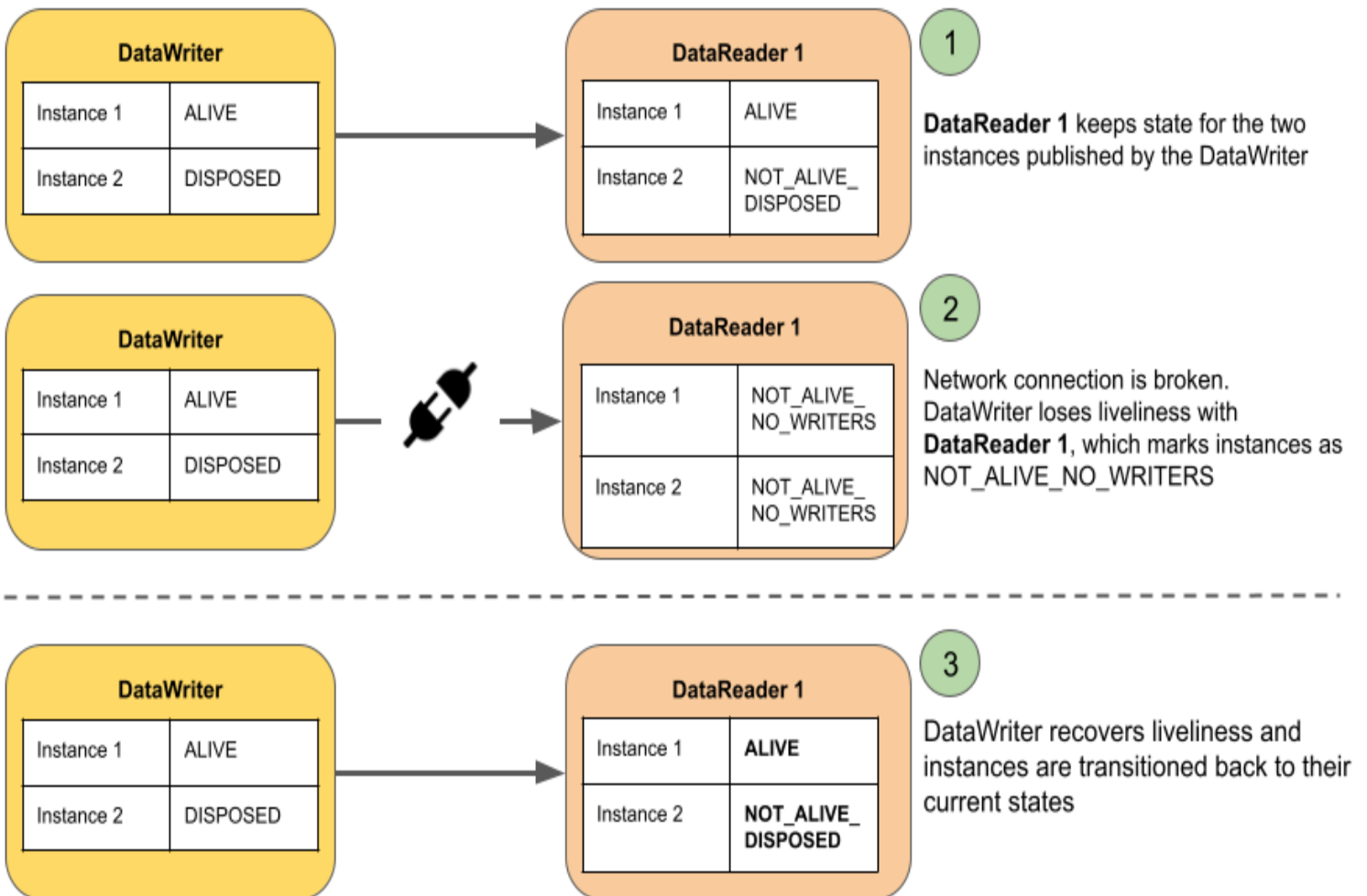
In previous releases, if the *DataReader* rediscovered a *DataWriter* previously publishing the instance, the state of the instance was not updated on the *DataReader* until the *DataWriter* sent a new sample of the instance. Instead, the instance would stay in the `NOT_ALIVE_NO_WRITERS` state. This could lead to inconsistent instance states: a late-joining *DataReader* (with a durability of `TRANSIENT_LOCAL` or higher) would have the correct instance state by virtue of discovery and receiving the samples from the *DataWriter*, whereas existing *DataReaders* would have the wrong state until they received a new sample. See [Figure 2.3: Behavior without Instance State Consistency Enabled on the next page](#).

Figure 2.3: Behavior without Instance State Consistency Enabled



In this release, *Connex* provides a new, experimental QoS setting, **instance_state_consistency_kind** (later renamed **instance_state_consistency_kind**), in the RELIABILITY QoS Policy, that enables a *DataReader* to update the instance state when a *DataWriter* is rediscovered without the *DataReader*'s having to receive samples for the instance. With instance state consistency enabled, when a *DataWriter* is rediscovered, the *DataReader* will send it a request for any instance state transitions that were missed while the two endpoints were disconnected. The *DataWriter* will send a response. The *DataReader* uses the information in the response to set the state of the instances to the same state they would have been in had the disconnection never occurred. Everything happens without any intervention from you, other than setting the QoS. Any instance state transitions that occur due to the *DataWriter*'s response sample are presented to the user application in the form of samples with the **valid_data** flag set to FALSE in the associated **SampleInfo**.

Figure 2.4: Behavior with Instance State Consistency Enabled



For details, see the the "Transition after NOT_ALIVE_NO_WRITERS" section in *Instance States*, in the [RTI Connex Core Libraries User's Manual](#).

Note: Instance state consistency currently does not work between *DataWriters* and *DataReaders* communicating through *Routing Service*. You should not implement instance state consistency in that case.

2.1.3 Simple Participant Discovery Protocol 2.0 Improvements

The following improvements have been made to the experimental Simple Participant Discovery Protocol (SPDP2) introduced in release 7.0.0 and described here: [2.2.1 Discovery Scalability and Troubleshooting on page 37](#). See also the *Simple Participant Discovery 2.0* section of the [RTI Connext Core Libraries User's Manual](#) for more information.

2.1.3.1 Improved bandwidth usage through optimal sending of bootstrap messages

There have been several improvements to bandwidth usage when using the experimental Simple Participant Discovery Protocol 2.0 (SPDP2).

Previously, when a *DomainParticipant* discovered a remote *DomainParticipant*, it continued to send bootstrap messages (DATA(Pb)) to any locators that belonged to the remote participant. But this is unnecessary, because the remote participant will drop any bootstrap messages it receives from a participant it has fully discovered.

In 7.1.0, a participant stops sending bootstrap messages to all unicast locators that belong to a remote participant once the participant has received the remote participant's full configuration in a configuration message (DATA(Pc)). This includes any unicast locators that are initial peers. The participant also removes any multicast locators that the remote participant is using that are not initial peers. The participant continues to send bootstrap messages to any multicast locators that the remote participant is using if they are also part of the initial peers list. This allows the participant to still discover any other participants with the same multicast receive address.

When the local participant receives updated locators from a remote participant after an IP mobility event, or the local participant removes the remote participant (for example after losing liveness or receiving a dispose message from the remote participant), the local participant will once again send bootstrap messages to any locators that belonged to the remote participant that were also initial peers.

These changes in behavior result in lower bandwidth usage, because participants send fewer bootstrap messages once a system is fully discovered and is at a steady state.

2.1.3.2 Differentiate between bootstrap messages and configuration messages more easily with new flag in ParticipantBuiltinTopicData

Previously, there was no way to tell whether a message received by a *DomainParticipant* contained the full configuration of the remote *DomainParticipant* or only the bootstrapping information. Now, if you are using SPDP2 (and have installed a listener on the ParticipantBuiltinTopicData builtin *DataReader* and set `ignore_default_domain_announcements` to FALSE—the default is TRUE), you will receive a

sample on the ParticipantBuiltinTopicData builtin *DataReader* both when the participant receives a bootstrap message and when the participant receives a configuration message.

A **partial_configuration** flag has been added to the ParticipantBuiltinTopicData. When this flag is TRUE, only the bootstrapping information in the sample is valid. When this flag is FALSE, the sample contains both the bootstrapping and configuration information of the remote participant. See the *Built-in DataReaders* section of the [RTI Connex Core Libraries User's Manual](#) for which fields are valid when this flag is set to TRUE.

2.1.3.3 Configure how configuration messages are sent using new QoS

In SPDP2, when two participants have received each other's bootstrap messages, they will exchange the rest of their configuration over a reliable channel using a configuration message. SPDP2 uses the ParticipantConfigBuiltinTopicData builtin *Topic* and associated builtin *DataWriter* and *DataReader* to send those configuration messages.

In 7.0.0, when SPDP2 was first released, the ParticipantConfigBuiltinTopicData builtin *DataWriter* and *DataReader* used the same QoS settings as the PublicationBuiltinTopicData builtin *DataWriter* and *DataReader*. Now, the ParticipantConfigBuiltinTopicData builtin *DataWriter* and *DataReader* have their own QoS settings. These QoS settings are used to tune the builtin *DataWriter* and *DataReader* that sends and receives the configuration messages.

The new QoS settings available in the DISCOVERY_CONFIG QosPolicy for the *DomainParticipant* are as follows:

- participant_configuration_writer
- participant_configuration_writer_data_lifecycle
- participant_configuration_writer_publish_mode
- participant_configuration_reader
- participant_configuration_reader_resource_limits

For details, see the *DISCOVERY_CONFIG QosPolicy (DDS Extension)* section of the [RTI Connex Core Libraries User's Manual](#).

Note: The ParticipantConfigBuiltinTopicData builtin *DataWriter* and *DataReader* are only enabled when using SPDP2.

2.1.3.4 remove_peer() API does not work with SPDP2 (known limitation)

The **remove_peer()** API is no longer supported when using SPDP2. Attempting to use **remove_peer()** will fail if called on a *DomainParticipant* using SPDP2.

2.1.4 API Improvements

2.1.4.1 Python API Improvements

2.1.4.1.1 Consistency and ease of use for read/take operations of IDL-based readers (dds.DataReader) and DynamicData readers

In previous releases, the read/take operations of IDL-based readers (**dds.DataReader**) and DynamicData readers (**dds.DynamicData.DataReader**) were not equivalent. This release makes the reader APIs equivalent with the following changes:

Changes to **dds.DynamicData.DataReader** and the builtin discovery readers (such as **dds.ParticipantBuiltinTopicData.DataReader**):

- The methods **read()** and **take()** now return copies of the data in a list of data/info named tuples, instead of loaned data and info in a **LoanedSamples** collection.
- The new methods **read_loaned()** and **take_loaned()** return loaned data and info in a **LoanedSamples** collection.
- The new methods **read_data()** and **take_data()** return copies of data in a list of data objects. The methods **read_valid()** and **take_valid()** have been removed.
- The new methods **take_async()** and **take_data_async()** provide copies of the data and data/info asynchronously.
- The methods **take_next()** and **read_next()** have been removed. Use **take()** and **read()** instead, and to read only one sample, use **select()** with a **max_samples** of 1.
- In most cases the versions that copy the data provide better performance than **read_loaned()** and **take_loaned()**, unless the data samples are very large.

Changes to **dds.DataReader**:

- The methods **read()** and **take()** continue to return copies, but they now return a list of named tuples, instead of a list of unnamed tuples, so the following code is now possible:

```
for sample in reader.take():
    data = sample.data
    info = sample.info
    do_something(data, info)
```

Note that this code continues to work as before:

```
for data, info in reader.take():
    do_something(data, info)
```

The same changes have been made to each *DataReader*'s Selector class.

2.1.4.1.2 XML-Based Application Creation now fully supported, with added support for use of IDL types

This release adds the ability to use Python classes (derived from IDL) as the topic-type in the XML definition of *DataReaders* and *DataWriters*. This functionality was missing in the Python API in the previous release. Adding this functionality completes support in the Python API for XML-Based Application Creation.

For more information, see the [API Reference: API Overview, XML Application Creation](#).

2.1.4.2 Topic-types can now be registered with a custom name

In previous releases, in the Python API the **dds.Topic** constructor didn't provide an argument to specify the name of the type to be registered. The actual name of the class was used. While using a different type name is usually not necessary, there are some situations where it is needed (for example, to inter-operate with applications using a different version of the type and the type definition can't be used to determine compatibility).

This release adds an extra optional argument to the Topic constructor that allows specifying the name to register the type with.

2.1.4.3 Set InitialObjectsPerThread QoS parameter in C# API

For completeness, InitialObjectsPerThread was added to SYSTEM_RESOURCE_LIMITS QoS Policy. This allows you to control the amount of memory initially allocated in each thread for thread-specific storage. However, the amount of thread-specific storage is now automatically managed by *Connex*, so it is very unlikely that you will need to set this parameter.

2.1.4.4 New non-throwing variants provided for some critical functions in Modern C++ API

Some applications need to ensure that their critical code paths do not allocate memory and therefore can't throw exceptions. For this reason, the *DataReader* **read()** and **take()** operations as well as the *DataWriter* **write()** operation now include variants with the **noexcept** guarantee:

Existing Function	New Function
<code>DataReader::take()</code> , <code>DataReader::read()</code>	<code>DataReader::take_noexcept()</code> , <code>DataReader::read_noexcept()</code>
<code>DataReader::Selector::take()</code> , <code>DataReader::Selector::read()</code>	<code>DataReader::Selector::take_noexcept()</code> , <code>DataReader::Selector::read_noexcept()</code>
<code>DataWriter::write()</code>	<code>DataWriter::write_noexcept()</code>

The new **noexcept** functions return a Result type, which contains the original return type plus a return code that provides information about the result of the operation. The error return codes in a Result object correspond to the exceptions that the throwing variants would throw in case of error.

See the API Reference for more information on each of these functions and on the new Result type.

2.1.5 Debuggability Improvements

2.1.5.1 Discovery snapshots now show domain tags and partition information

Discovery snapshots readability has been improved, and now they show information about domain tags and partitions. See the *Discovery Snapshots* section of the [RTI Connex Core Libraries User's Manual](#) for more information.

2.1.5.2 Debug root cause of file open and close operations easily with enhanced log messages

File opening/closing error messages have been updated to contain information about the root cause of the error.

Previously, the error message when an open operation failed was as follows:

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

Now opening and closing error messages will follow this format:

```
OPEN FILE FAILURE | example.xml with error code 2 (No such file or directory) in DDS_
XMLFileInfoList_assertFile
```

2.1.5.3 Manually enable or disable logging backtrace upon SIGSEGV signal from a Connex application, using new property

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

In this release, a new property, **dds.participant.enable_backtrace_upon_sigsegv**, allows you to manually enable or disable the use of this logging feature. (See the "PROPERTY QosPolicy (DDS Extension)" section in the [RTI Connex DDS Core Libraries User's Manual](#).) By default, logging of the backtrace will be disabled in release libraries, but enabled in debug libraries. See "Potential hang upon SIGSEGV signal from a *Connex* application" (CORE-12794) in the [RTI Core Libraries Release Notes](#).

The accepted values for the new property are "auto" for the default behavior (backtrace enabled only in debug libraries), "true" for enabling the logging of the backtrace in both debug and release libraries, and "false" for disabling it in both release and debug libraries.

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

2.1.5.4 More easily debug failures due to DestinationOrder by source timestamp mismatches, with enhanced log messages

When **addSample** fails due to a DestinationOrder by source timestamp mismatch, *Connex* now logs the two timestamps that were compared. This message is generated at the warning logging level.

Additionally, timestamp comparison logs now display the timestamps at a microsecond resolution.

2.1.6 Platform and Build Changes

2.1.6.1 Support for .NET/C# API on macOS Systems with Arm v8 CPUs

The Connex .NET/C# API is now supported on macOS® systems with Arm® v8 CPUs. Previously, the .NET/C# API was only supported on macOS systems with x64 CPUs.

2.1.6.2 New Platforms

RTI has validated that the *Connex* libraries for Red Hat® Enterprise Linux 8 systems (x64Linux4gcc7.3.0) also work on Red Hat Enterprise Linux 9 systems.

The architecture x64Darwin17clang9.0 has been replaced with x64Darwin20clang12.0.

Table 1 New Platforms

OS	OS Version	CPU	Toolchain	RTI Architecture
Linux	Red Hat Enterprise Linux 9	x64	gcc 7.3.0	x64Linux4gcc7.3.0
macOS	macOS 11, 12	x64	clang 12.0, 13.0	x64Darwin20clang12.0

2.1.6.3 Removed Platforms

This release removes support for the following platforms. Note that newer versions of macOS (macOS 11 and 12) are supported. Support for VxWorks will return in a future release.

Table 2 Removed Platforms

OS	OS Version	CPU	Toolchain	RTI Architecture
macOS	macOS 10.13-10.15	x64	clang 9.0, 10.0, 11.0	x64Darwin17clang9.0
VxWorks	VxWorks 21.11	x64	llvm 12.0.1.1	x64Vx21.11llvm12.0.1.1 x64Vx21.11llvm12.0.1.1_rtp

2.1.6.4 Added QNX7.1 support to FindRTIConnexDDS script

The "Find Package" script (**FindRTIConnexDDS.cmake**) now supports building the arm-v8QNX7.1qcc_gpp8.3.0 architecture.

2.1.6.5 New FindPackage component allows building the Security Plugins against wolfSSL

A new component, **security_plugins_wolfssl**, has been added to the "Find Package" script (**FindRTIConnexDDS.cmake**) in order to link the *RTI Security Plugins* library against wolfSSL®. The component **security_plugin** continues to link against OpenSSL®.

2.1.6.6 New components and imported target libraries for Cloud Discovery Service, Persistence Service and Web Integration Service added to "FindPackage" CMake script

As mentioned in their release notes, *Cloud Discovery Service*, *Persistence Service*, and *Web Integration Service* are now installed with target libraries. Those target libraries have been added to the "Find Package" (FindRTIConnexDDDS.cmake) script:

- **cloud_discovery_service** components:
 - **RTIConnexDDDS::cloud_discovery_service_c**
 - **RTIConnexDDDS::cloud_discovery_service_cpp2**
- **persistence_service** component:
 - **RTIConnexDDDS::persistence_service_c**
- **web_integration_service** component:
 - **RTIConnexDDDS::web_integration_service_cpp2**

2.1.7 Deprecations and Removals

This section describes products, features, and platforms that are *deprecated* or *removed* starting in release 7.1.0.

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.

2.1.7.1 Monitoring Library deprecated in this release

In this release, RTI announces the deprecation of the RTI Monitoring Library (described in the *RTI Monitoring Library* section of the [RTI Connex Core Libraries User's Manual](#)) and *RTI Monitor*. RTI is introducing a new Connex Observability Framework (see [2.1.1 Monitor Health of Connex Applications Using New Connex Observability Framework \(Experimental\)](#) on page 19); in the future, this new framework will replace the current Monitoring Library.

If you want to start monitoring your deployed systems using a monitoring API, you should use the new library; if you're already using the current Monitoring Library API and plan to move to future *Connex* releases, you should plan your migration now. RTI will continue to support the current Monitoring Library and *Monitor* for customers using 6.1.x. This deprecation notice applies to customers using 7.1 and later.

2.1.7.2 Removed ability to share a database connection in Persistence Service and durable writer history

This release removes the ability to share a database connection in *RTI Persistence Service* (which is done by setting the tag `<share_database_connection>` to true for a `<persistence_group>`). It also removes the ability to share a database connection when using durable writer history and setting the property `dds.data_writer.history.odbc_plugin.builtin.shared` to 1.

Note that sharing a database connection was only allowed for external databases, and support for external databases was removed in 7.0.0 (see [2.2.12.4 Durable writer history, durable reader state, and Persistence Service no longer support external databases on page 62](#)).

2.1.8 Product Availability Changes

2.1.8.1 RTI Limited Bandwidth Endpoint Discovery now installed with RTI Connex Professional

Starting in 7.1.0, *RTI Limited Bandwidth Endpoint Discovery (LBED)* is installed with the *RTI Connex Professional* bundle. You no longer need to purchase and install LBED separately. It is installed with your `rti_connex_dds-7.1.0-pro-<host or target>-<host platform or target architecture>.<extension or rtipkg>` bundle. Because of this change, LBED is also no longer included with the *RTI Limited Bandwidth Plugins*. See the *Limited Bandwidth Endpoint Discovery* section of the [RTI Connex Core Libraries User's Manual](#) and the [RTI Limited Bandwidth Endpoint Discovery User's Manual](#).

2.1.8.2 RTI Web Integration Service now installed with RTI Connex Professional

Starting in 7.1.0, *RTI Web Integration Service* is installed with the *RTI Connex Professional* bundle. (It was always included with *Professional*, but you had to install it separately.) You no longer need to install *Web Integration Service* separately. It is installed with your `rti_connex_dds-7.2.0-pro-<host or target>-<host platform or target architecture>.<extension or rtipkg>` bundle. For more information, see the documentation for *Web Integration Service* in `<NDDSHOME>/doc/manuals/connex_dds_professional/services/web_integration_service`. See the Installation chapter.

2.1.8.3 Cloud Discovery Service now shipped as host + target, consistent with other Infrastructure Services, by providing target-specific libraries/executables

RTI Cloud Discovery Service is now shipped in two parts, as a host and a target package. This change brings *Cloud Discovery Service* in line with how other *Connex* packages are structured and ensures

that you have target-specific libraries when using the service-as-a-library functionality. See the [RTI Cloud Discovery Service](#) documentation for more information.

2.1.8.4 TLS Support now included with Connex Secure and Connex Anywhere

In release 7.1.0, *RTI TLS Support* is now included with the purchase of the *Connex Secure* and *Connex Anywhere* bundles. It is still installed separately. See the *RTI TLS Support Installation Guide*.

2.1.9 Other

2.1.9.1 Connex Now Searches for Included XML Configuration Files in Directory where File is Located

This release improves the way *Connex* searches for the XML contained in the file attribute of the `<include>` XML tag.

In previous releases, *Connex* only searched the current working directory for XML files. For example, assume these three files:

File1.xml (located in the current working directory)

```
<dds>
  <types>
    <include file="./nested/File2.xml"/>
    ...
  </types>
</dds>
```

nested/File2.xml

```
<dds>
  <types>
    <include file="File3.xml"/>
    ...
  </types>
</dds>
```

nested/File3.xml

```
<dds>
  <types>
    ...
  </types>
</dds>
```

Previously, *Connex* could not parse **File2.xml**, because in **File2.xml**, the path to **File3.xml** is given as *relative to File2.xml's* current directory, but the parser wasn't looking in that directory for **File3.xml**; it was looking only in the current working directory (where **File1.xml** is) for **File3.xml**. In this release, *Connex* now also considers the directory containing the current file being parsed (**File2.xml**) to resolve the path for its included files (**File3.xml**).

The search order for included files is now as follows:

1. **(new step)** *Connex*t searches for the included files contained in a file in the directory where that file is located.
2. *Connex*t searches for the included files in the current working directory.

Currently, this feature is only supported on the following systems, for the platforms listed in the Supported Platforms section of the [RTI Connex Core Libraries Release Notes](#):

- AIX®
- Android™
- Linux® (except for POSIX®-compliant platforms for *RTI Connex TSS*)
- macOS®
- Windows®
- QNX®

2.1.9.2 Connex Now Uses RTPS 2.5

RTI has updated the RTPS wire protocol version number that *Connex*t announces in messages it puts on the wire, from version 2.3. to version 2.5. This is a result of RTI's work towards supporting the RTPS protocol with RTPS version 2.5 features. For details, see *Compatibility*, in the [RTI Core Libraries Release Notes](#).









2.2 What's New in 7.0.0

RTI® Connex® 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 Connex 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 2.1 Release 7.0.0 Highlights (these and more features are described below)

	<p>Ability to partition groups of applications</p> <p>Now create groups of applications at the <i>DomainParticipant</i> level to reduce network traffic, CPU, and memory utilization.</p>
	<p>DDS Spy and DDS Ping Improvements</p> <p>Debug applications from the command line faster with improved <i>DDS Spy</i> output. Use filters to display discovery data, user data, or everything.</p>
	<p>DDS-XML format for Limited Bandwidth Endpoint Discovery Plugin</p> <p>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.</p>
	<p>Faster Debugging with New Logging Categories</p> <p>Two new logging categories, for discovery and security activity, enable you to more easily filter and trace discovery operations and <i>RTI Security Plugins</i> operations. See Security Improvements and Logging Improvements.</p>
	<p>Reduce network bandwidth utilization by allowing Connex to limit allowed interfaces even further</p> <p>A new <code>max_interface_count</code> field limits the number of network interfaces used by a <i>DomainParticipant</i>.</p>
	<p>Reorganized Core Libraries User's Manual for easier browsing</p> <p>A reorganized <i>RTI Connex Core Libraries User's Manual</i> is easier to use for people who are new to the Connex ecosystem.</p>
	<p>Enhanced scalability for peer-to-peer communications with new Simple Participant Discovery Protocol 2.0 (experimental)</p> <p>For systems with a large number of <i>DomainParticipants</i>, the SPDP 2.0 option improves scalability by sending optimized matching and liveliness messages among <i>DomainParticipants</i>.</p>
	<p>Connex Python API adds support for user data types and code generation (experimental)</p> <p>Get started with Python faster for your application's data types. Now use <i>Code Generator</i> to generate code in Python.</p>

For what's new and fixed in other products included in the *Connex* 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 [2.2.13 Product Availability Changes on page 64](#).

2.2.1 Discovery Scalability and Troubleshooting

2.2.1.1 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 liveness. 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 liveness.

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 Connex Core Libraries User's Manual](#)) `SPDP2 | SEDP`. This will enable SPDP 2.0 and the existing Simple Endpoint Discovery Protocol.

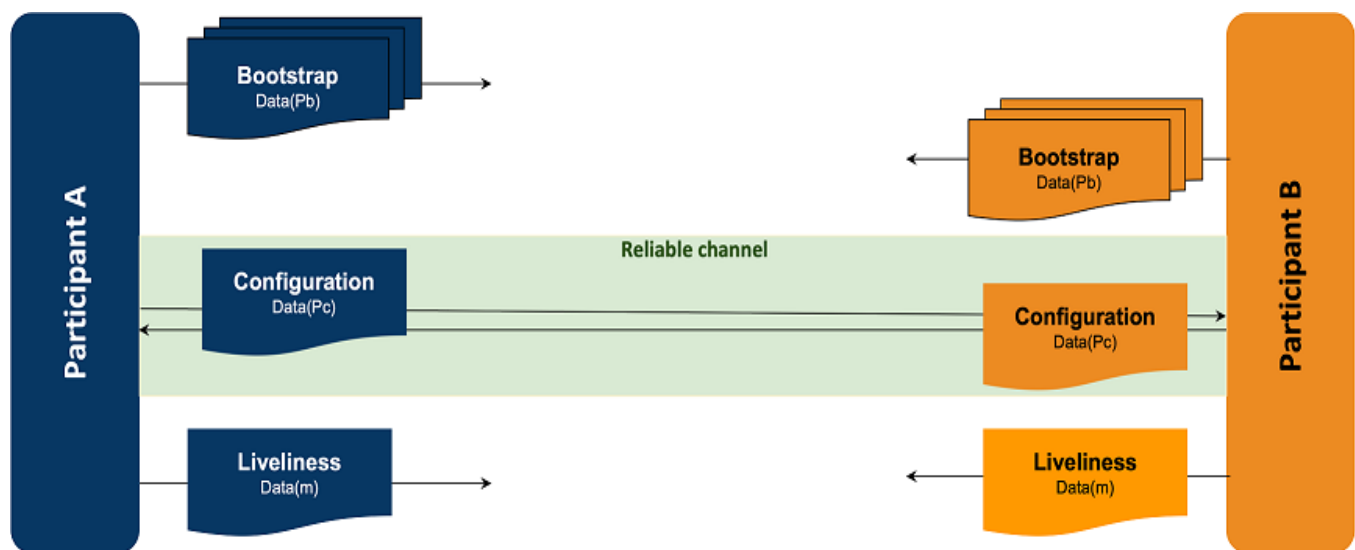
In the current SPDP, participants send out participant announcements (also known as participant DATA submessages) that contain all of the information that a participant needs to communicate with remote

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 2.5: 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 *Simple Participant Discovery 2.0* section of the [RTI Connext Core Libraries User's Manual](#) for further details.

2.2.1.2 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.

2.2.1.3 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 *Connex* 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="udp4://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.

2.2.1.4 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](#).

2.2.2 Language Bindings, APIs, XML Configuration

2.2.2.1 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 data-classes.

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

2.2.2.2 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](https://community.rti.com/documentation) on the RTI Community Portal (<https://community.rti.com/documentation>) and [this code example](#).

2.2.2.3 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.

2.2.2.4 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.

2.2.2.5 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 *Connexrt* 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);
});
```

2.2.2.6 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 Connex 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 Connex 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.

2.2.2.7 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
```

2.2.2.8 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**.

2.2.2.9 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 *Connex* releases. When `must_interpret` is set to `false`, only the versions of *Connex* 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 *Connex* prior to 7.0.0. For more information, read the *How the XML is Validated* section of the [RTI Connex Core Libraries User's Manual](#).

2.2.3 Usability

2.2.3.1 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*.

2.2.3.2 Reorganized Core Libraries User's Manual for easier browsing

The *RTI Connex 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 2.6: Improved User's Manual Structure

Part 1: Connex Overview	▼	
Part 2: Building Blocks of Connex: Entities and Domains	▼	This content comes earlier
Part 3: Working with Data in Connex	▲	
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	▲	This content comes earlier and is no longer buried under Advanced Concepts
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—Ignores Entities	▼	
28. Accessing Discovery Information through Built-In Topics	▼	
Part 5: Sending Data with Connex	▼	This content is no longer buried under Core Concepts
Part 6: Receiving Data with Connex	▼	
Part 7: Configuring Connex Using QoS	▼	QoS Policies are now all in one section, rather than dispersed throughout the manual
Part 8: Working with Transports in Connex	▼	
Part 9: Debugging and Monitoring Connex Applications	▼	
Part 10: Alternative Communication Models	▼	
Part 11: Connex DDS Threading Model	▼	... and other improvements

2.2.3.3 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 *Connex Professional*, *Connex Secure*, and *Connex Anywhere*. 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 tables in the RTI Connex Core Libraries Release Notes](#).

2.2.3.4 System resources easier to manage with `max_objects_per_thread` now configured by Connexrt 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, *Connexrt* 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, *Connexrt* 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 *Connexrt* is initialized. This was the default behavior in previous releases.

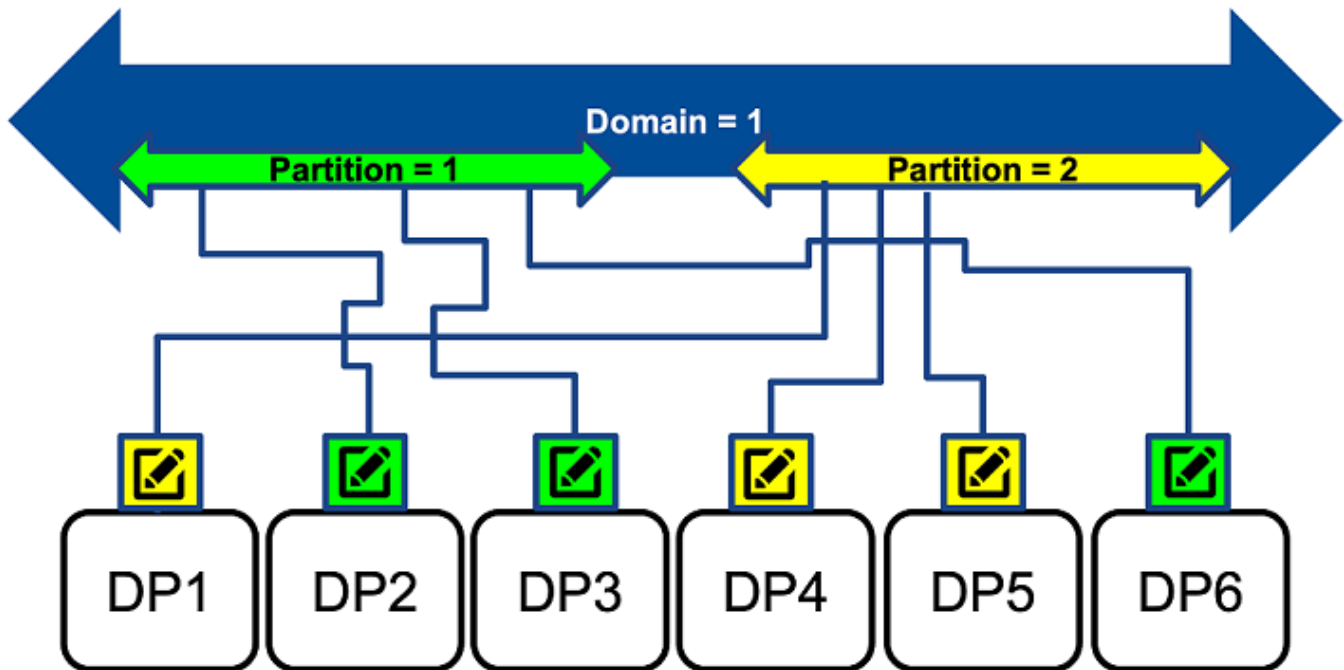
See the *SYSTEM_RESOURCE_LIMITS QoS Policy (DDS Extension)* section of the [RTI Connexrt Core Libraries User's Manual](#) for more information.

2.2.4 Scalability

2.2.4.1 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 QoS Policies). 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 2.7: 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 QoS Policy* section of the [RTI Connext Core Libraries User's Manual](#) for more information.

2.2.5 Bandwidth Sensitivity

2.2.5.1 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.

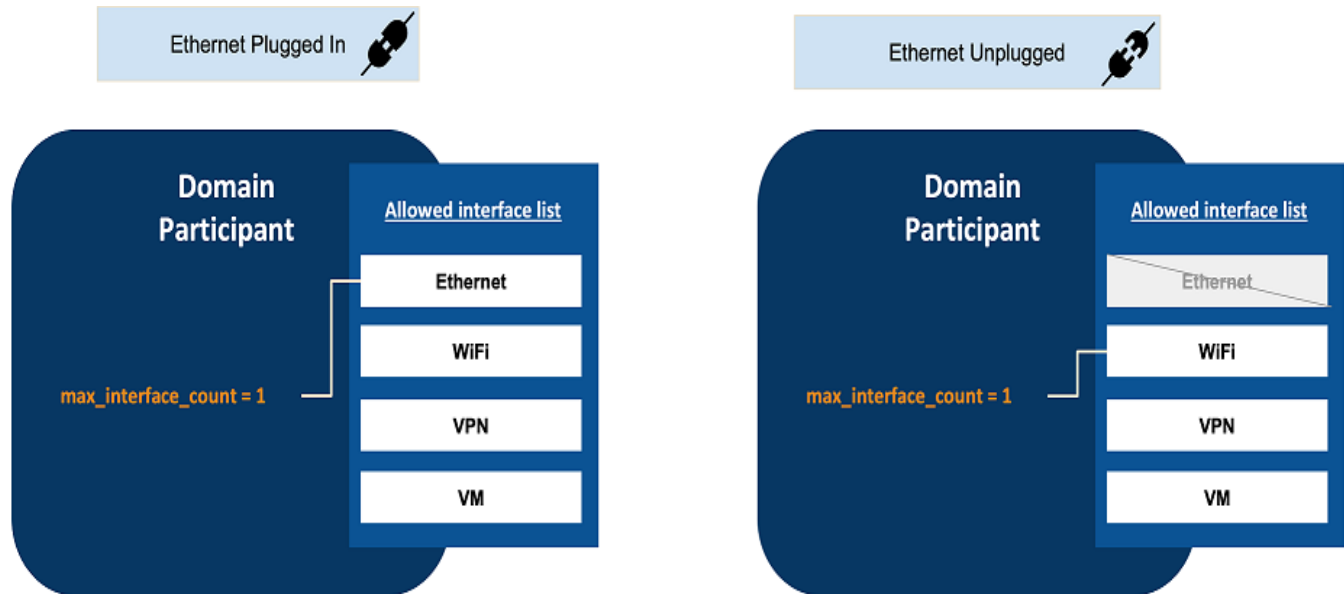
If you are not using the TopicQuery or locator reachability features and would like to avoid incremental network traffic, you can disable the creation of the ServiceRequest channel with a new *DomainParticipant* QoS setting: **enabled_builtin_channels** (part of the DISCOVERY_CONFIG QoS Policy). Please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

2.2.5.2 Reduce network bandwidth utilization by allowing Connex 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 2.8: 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](#).

2.2.5.3 More efficient bandwidth utilization for configurations with small transport message_size_max

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

Reducing the *Connex*t 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, *Connex*t 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, *Connex*t could not send samples.

Starting with release 7.0.0, the calculation of the RTPS protocol overhead is automatically done by *Connex*t 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.

2.2.6 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*.

2.2.6.1 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 Connex Logging* section of the [RTI Connex 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.

2.2.6.2 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 QoS Policy* section of the [RTI Connex Core Libraries User's Manual](#) for more details on compression.

2.2.7 Logging Improvements

2.2.7.1 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 Connex Logging* section of the [RTI Connex Core Libraries User's Manual](#).

2.2.7.2 Improved logging in specific scenarios

2.2.7.2.1 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.

2.2.7.2.2 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
```

2.2.7.2.3 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. *Connex*, however, will log a warning if there are no reachable locators for a remote entity, with the following message:

```
COMMENTBcWriterService_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.
```

2.2.7.2.4 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)
```

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

When *Connex* 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).
```

2.2.7.2.6 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'
```

2.2.7.3 Changes to log message verbosity

2.2.7.3.1 "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 Connex Logging* section of the [RTI Connex Core Libraries User's Manual](#).

2.2.7.3.2 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 Connex Logging* section of the [RTI Connex Core Libraries User's Manual](#) for more information on verbosity levels.

2.2.7.4 Changes to Activity Context in log messages

In release 6.1.0, *Connex* 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 Connex Core Libraries User's Manual](#).

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

2.2.7.4.1 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]
```

2.2.7.4.2 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}]
```

2.2.7.4.3 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 *Connex* fails to send an ACK, described in the following error message: “!send periodic ACK”. Now, *Connex* 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
```

2.2.7.5 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 Connex API Reference > General Utilities > Heap Monitoring > take_snapshot**).

2.2.8 DDS Spy and DDS Ping Improvements

2.2.8.1 Debug Connex 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](#).

2.2.8.1.1 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")
```

2.2.8.1.2 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
shapessize: 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, "shapetype":30, "fillKind":"SOLID_
FILL", "angle":0}
```

2.2.8.1.3 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"
```

2.2.8.1.4 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"
```

2.2.8.2 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](#).

2.2.8.3 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:

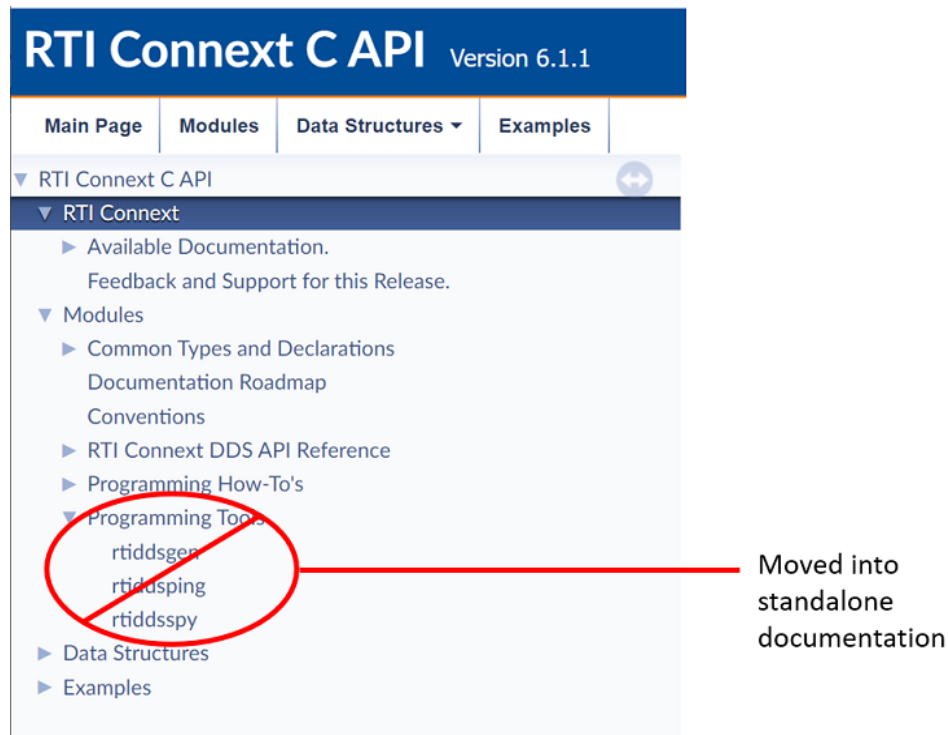
```
13:29:20 New writer      from 192.168.42.107 : topic="Example test" type="test"
partition="P1"
13:29:22 New data       from 192.168.42.107 : topic="Example test" type="test"
partition="P1"
13:29:23 Updated writer  from 192.168.55.133 : topic="Example test" type="test"
partition="P2"
13:29:24 New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

2.2.8.4 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 2.9: Moved Documentation



2.2.9 Platform and Build Changes

2.2.9.1 Dynamically Load Monitoring Library and Security Plugins on VxWorks

Connex 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 Connex 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.

2.2.9.2 New Supported Platforms

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

Table 3 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

OS	OS Version	CPU	Toolchain	RTI Architecture
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

2.2.10 Changes to Defaults

2.2.10.1 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.

2.2.10.2 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 [2.2.3.4 System resources easier to manage with max_objects_per_thread now configured by Connexx automatically on page 46](#) for more information.

2.2.10.3 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 Connexx 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.

2.2.11 Performance Improvements

2.2.11.1 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*.

2.2.11.2 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.

2.2.11.3 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.

2.2.12 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.

2.2.12.1 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.

2.2.12.2 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 7.2.0. 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 7.2.0. (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](#).

2.2.12.3 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.

2.2.12.4 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.

2.2.12.5 RTI Secure WAN Transport removed in this release

Secure WAN Transport—which includes both the Secure WAN Transport and Secure Transport, both using DTLS—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 Connex Core Libraries User's Manual](#) for more information.

2.2.12.6 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.

2.2.12.7 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.

RTI decided to invest in adding data-publishing capability to *Administration Console*, rather than maintaining *Spreadsheet Add-in for Excel*. An integrated solution in *Administration Console* will be easier to use. It will also be available to all *Connex* users on most platforms. Not all users have access to Excel and RTI only supported it on Windows.

2.2.12.8 RTI Connector for Python deprecated in this release

With the introduction of the [Connex 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 [2.2.2.1 Connex Python API adds support for user data types and code generation \(experimental\) on page 40](#)).

2.2.12.9 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

2.2.12.10 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.)

2.2.12.11 Deprecated `rtps_overhead` Property

The property `dds.participant.protocol.rtps_overhead` has been deprecated, since there is no need to use it anymore. See [2.2.5.3 More efficient bandwidth utilization for configurations with small transport message_size_max on page 50](#) for an explanation.

2.2.13 Product Availability Changes

2.2.13.1 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 Connex Core Libraries Release Notes](#) for information on RTI experimental products and features.

2.2.13.2 RTI Database Integration Service

Database Integration Service is not included in release 7.0.0.

2.2.13.3 RTI Limited Bandwidth Endpoint Discovery Plugin

In release 7.0.0, the new and improved *RTI Limited Bandwidth Endpoint Discovery Plugin* (LBED) (see [2.2.3.1 DDS-XML format for Limited Bandwidth Endpoint Discovery Plugin on page 43](#)) is now

included with the purchases of some bundles, including *Connex 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 *Connex* 6.1.1 LTS or earlier with *RTI Limited bandwidth Plugins* (an add-on product). The LBED enhancements in release 7.0.0 (see [2.2.3.1 DDS-XML format for Limited Bandwidth Endpoint Discovery Plugin on page 43](#)) will be available in a future official product release.

2.2.14 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 *Connex* products, see the "3rdPartySoftware" documents in your installation: `<NDDSHOME>/doc/manuals/connex_dds_professional/release_notes_3rdparty`.