RTI Queuing Service

Release Notes

Version 5.3.1

Your systems. Working as one.

Printed in U.S.A. First printing.
February 2018.

Trademarks

Copy and Use Restrictions

Technical Support

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

Contents

Release Notes

1 Supported Platforms

RTI® Queuing Service is supported on the platforms in Table 1.1.

Table 1.1 **Supported Platforms**

Platform	Operating System
Linux® (Intel® CPU)	CentOS 6.0, 6.2 - 6.4, 7.0
	Red Hat® Enterprise Linux 6.0 - 6.5, 6.7, 6.8, 7.0
	Ubuntu 12.04 LTS, 14.04 LTS, 16.04 LTS
OS X®	All OS X platforms listed in the *RTI Core Libraries Platform Notes*.
Windows®	All Windows platforms listed in the *RTI Core Libraries Platform Notes*.

2 Compatibility

Queuing Service is built on top of, and intended for use with, *RTI Connext® DDS* with the same version number.

3 What's Fixed in 5.3.1

This section describes bugs fixed in 5.3.1. These fixes have been made since 5.3.0.

3.1 Missing Library needed to Link Queuing Service C++ API Dynamically on Windows Systems

The file **rticonnextmsgcpp2.lib** was not included and therefore it was not possible to link the Queuing Service C++ API dynamically on Windows platforms. The file **rticonnextmsgcpp2z.lib**, which allows static linking, was available. This problem has been resolved; now both the dynamic and static libraries are provided.

[RTI Issue ID QUEUEING-665]

4 Previous Release

4.1 What's New in 5.3.0

This section highlights new platforms and improvements in 5.3.0. These enhancements have been made since 5.2.3.

4.1.1 New Platforms

This release adds support for platforms on the following operating systems:

- OS X 10.12
- Red Hat Enterprise Linux 6.8
- Ubuntu 16.04 LTS
- Windows Server 2016

For details, see the *RTI Core Libraries Platform Notes*.

4.1.2 Platforms on Legacy Operating Systems

The following legacy operating systems have reached end-of-life from their corresponding vendors. Please contact RTI support or your account manager if you require version 5.3 to run on these platforms:

- Red Hat Enterprise Linux 5.0

4.1.3 Removed Platforms

Platforms on the following operating systems are no longer supported:

- OS X 10.8
- Windows 2003, Windows Vista, Windows XP

4.1.4 Timed-Out Samples now Redelivered to Same Consumer, even if Another Consumer is Available

The last sample that timed out for a consumer will NOT be sent to the same consumer if there is another consumer available.

4.1.5 Rejected Samples Now Sent to Alternative Consumer when Possible

Queuing Service will avoid redelivering the last sample rejected by a consumer to the same consumer if it can find another available consumer.

4.1.6 Modern C++ API for Queuing Service

The modern C++ wrapper API simplifies the interaction of queue producers and queue consumers with *Queuing Service*.

This API implements the following abstractions:

- QueueProducer
- QueueConsumer
- QueueRequester
- QueueReplier

The *Queuing Service* wrapper API is also available for .NET.

For more information, see *Queuing Service User's Manual* and the Connext DDS API Reference HTML documentation.

4.1.7 Queuing Service Default Configuration was Suboptimal for More than 32 Consumers

If you tried running *Queuing Service* with more than 32 consumers, you may have noticed a significant slow down and performance drop.

You could have resolved this problem by setting the following for your shared reader in the Queuing Service XML configuration:

```
<datawriter_qos>
    <writer_resource_limits>
        <max_remote_reader_filters>
            DDS_LENGTH_UNLIMITED
        </max_remote_reader_filters>
    </writer_resource_limits>
</datawriter_qos>
```

You no longer need to set **max_remote_reader_filters** as seen above, because now it is part of the default Queuing Service configuration.

4.1.8 Support for Remote ShutDown

The remote administration functionality has been extended with a new command that allows you to shut down the service. A service receiving the shutdown command will finish the execution and the process will exit. This command is applicable if you run the service from the supplied executable included in the *Connext DDS* distribution.

4.1.9 Support for Native Heap Monitoring

Queuing Service incorporates a native heap memory monitor that allows you to analyze the allocations performed at the service and *RTI Connext DDS* layers. You can use heap monitoring through the command line with the following options:

-heapSnapshotPeriod: <sec> Enables heap monitoring. Generate heap snapshot every <sec>.

-heapSnapshotDir: <dir>> Output directory where the heap monitoring snapshot are dumped. The filenames of the generated dump files have the following format:

RTI_heap_<appName>_<processId>_<index>.log

where <appName> is the name you assigned to the service execution through the **-appName** parameter, <processId> is the process ID of the service execution, and <index> is an integer that automatically increases each snapshot period.

For details related to the format of the snapshot files see the API Reference HTML documentation for *Connext DDS*.

4.1.10 Support for User-Defined Sample Flags Propagation

Queuing Service has been enhanced to propagate the SampleFlag bits of the input samples. Only the user-defined sample flags are propagated.

4.2 What's Fixed in 5.3.0

This section describes bugs fixed in 5.3.0. These fixes have been made since 5.2.3.

4.2.1 Consumers not Discovered if Filter Expression Contained Extra Parenthesis

Consumers were not properly detected if their filter expression contained extra parentheses, such as:

(@related_reader_guid.value = &hex(00000000000000000000000000000007))

or

@related_reader_guid.value = &hex((00000000000000000000000000000007))

This problem has been resolved and consumers are discovered regardless of extra parentheses.

[RTI Issue ID QUEUEING-538]

4.2.2 DDS_REDELIVERED_SAMPLE Flag Set Incorrectly when Sample Rejected by Consumer

The *Queuing Service User's Manual* states that the DDS_REDELIVERED_SAMPLE flag is set for a sample if *Queuing Service* does not receive a consumer acknowledgment (AppAck) before a timeout period.

However, the flag was set each time a message was sent more than once, even if the message was negatively acknowledged by a consumer.

This problem has been resolved. Changes have been made to conform with the definition in the *User's Manual*. Now a message will be flagged if and only if a consumer response times out.

[RTI Issue ID QUEUEING-543]

4.2.3 failed_delivery_message_count not Properly Computed

The *Queuing Service User's Manual* defines a FailedDelivery as a sample that has not been successfully delivered to any Queue Consumer after the maximum number of attempts.

Inconsistently, **failed_delivery_message_count** counted the total number of failed attempts to send any message. For a given message, multiple failed attempts may occur before the message is discarded and becomes a FailedDelivery.

Changes have been made to make **failed_delivery_message_count** conform with the definition in the *User's Manual*. The counter increases only when a message is not successfully delivered despite having been sent the maximum number of times allowed (1 + **max_delivery_retries**).

[RTI Issue ID QUEUEING-563]

4.2.4 Queuing Service did not Discover Already Present Remote Administration Clients

Queuing Service did not discover remote administration clients that were present before the service was started. This problem has been resolved.

[RTI Issue ID QUEUEING-612]

4.2.5 Service Crashed if a Sample Selector Expression Contained an Empty String as Predicate

Queuing Service may have crashed if it received a remote command that accepted a sample selector expression containing an empty string as a predicate, such as "x = 1 OR", "OR", etc. This problem has been resolved and the service will send a reply indicating an error.

[RTI Issue ID QUEUEING-613]

4.2.6 Potential Segmentation Fault when Shutting Down Queuing Service

While using replication, you may have seen a segmentation fault in Queuing Service during the shutdown operation. This problem has been resolved.

[RTI Issue ID QUEUEING-628]

4.2.7 Memory Leak when Using a Persisted Service Configuration

If you ran Valgrind on a *Queuing Service* instance configured from a database file, you may have seen that after the instance finalized some memory, it was still not freed. This problem has been resolved.

[RTI Issue ID QUEUEING-634]

4.2.8 Configuration Validation Failed if Version Attribute Contained a Different Version Value

Queuing Service would fail to load a configuration that contained the **version** attribute, within <dds>, with a value different than the actual version of the service. For instance, if you ran Queuing Service 5.2.7 and the attribute value was 5.2.6, the validation would fail.

This problem was due to the XSD definition of the value attribute was FIXED to the service version. The FIXED modifier has been removed to resolve this problem.

[RTI Issue ID QUEUEING-653]

4.2.9 Potential Deadlock Stopping the Service and Remote Delete Queue Command is Received

Queuing Service may have entered in deadlock if monitoring was enabled and a remote command to delete a queue was received while the service was stopping. This problem has been resolved.

[RTI Issue ID QUEUEING-659]

---

5 Current Limitations

- The QueueProducer and QueueConsumer wrapper APIs are only supported for the Modern C++ and .NET APIs.

---

6 Available Documentation

Queuing Service documentation also includes:

- **Getting Started Guide** (RTI_Queuing_Service_GettingStarted.pdf)—Provides installation and startup instructions.
- **User's Manual** (RTI_Queuing_Service_UsersManual.pdf)—Describes how to configure and use *Queuing Service*.