RTI Data Distribution Service

Core Libraries and Utilities

What's New

in Version 4.5e

The Global Leader in DDS

Technical Support

Real-Time Innovations, Inc.
385 Moffett Park Drive
Sunnyvale, CA 94089

Phone:	(408) 990-7444
Email:	support@rti.com
Website:	https://support.rti.com/

Contents

What's New

This document highlights new or changed features in *RTI*® *Data Distribution Service* 4.5e. (For details on fixed bugs, please see the *Release Notes*.)

For more information, visit the RTI Knowledge Base, accessible from http://www.rti.com/support, to see sample code, general information on *RTI Data Distribution Service*, performance information, troubleshooting tips, and technical details. By its very nature, the knowledge base is continuously evolving and improving. We hope that you will find it helpful. If there are questions that you would like to see addressed or comments you would like to share, please send e-mail to **support@rti.com**. We can only guarantee a response to customers with a current maintenance contract or subscription. You can purchase a maintenance contract or subscription by contacting your local RTI representative (see http://www.rti.com/company/contact.html), sending an e-mail request to **sales@rti.com**, or calling +1 (408) 990-7400.

1 Group Ordered Access

This release introduces support for group ordered access. With this feature, *RTI Data Distribution Service* supports the delivery of data samples in the same order in which they were published by different *DataWriters* belonging to the same *Publisher*.

To request this behavior, the PresentationQosPolicy on both the *Publisher* and *Subscriber* must be set such that **access_scope** = GROUP_PRESENTATION_QOS and **ordered_access** = true.

Please refer to the *User's Manual* or HTML documentation for details on the following related changes:

❑ APIs

- The *Subscriber's* **begin_access()** and **end_access()** operations are now supported.

- The *Subscriber's* **get_datareaders()** operation now returns a list of *DataReaders* that indicates the order in which the application must read or take samples to get the samples in order.

❑ DataReaderQoS
- New field: **availability**
- New field: **subscription_name**

❑ DataWriterQoS
- New field: **publication_name**

❑ QoS Policies
- AvailabilityQosPolicy: New policy, applies to *DataReaders*
- EntityNameQosPolicy
    - New field: **role_name**
- PresentationQosPolicy
    - New PresentationAccessScopeKind: HIGHEST_OFFERED_PRESENTATION_QOS
- DataWriterProtocolQosPolicy's RtpsReliableWriterProtocol_t:
    - New field: **virtual_heartbeat_period**
    - New field: **samples_per_virtual_heartbeat**
- DataReaderResourceLimitsQosPolicy:
    - New field: **max_remote_virtual_writers**
    - New field: **max_remote_virtual_writers_per_sample**
- DataWriterResourceLimitsQosPolicy:
    - New field: **initial_virtual_writers**
    - New field: **max_virtual_writers**
- DomainParticipantResourceLimitsQosPolicy:
    - New field: **max_endpoint_groups**
    - New field: **max_endpoint_group_cumulative_characters**

❑ SampleInfo Structure
- New field: **original_publication_virtual_guid**
- New field: **original_publication_virtual_sequence_number**

❑ Status
- New reasons in SampleRejectedStatusKind: REJECTED_BY_VIRTUAL_WRITERS_LIMIT, REJECTED_BY_REMOTE_WRITERS_PER_SAMPLE_LIMIT

2 Prioritized Samples

This release introduces *Prioritized Samples,* a feature that allows you to prioritize traffic that is in competition for transmission resources. The granularity of this prioritization may be by *DataWriter*, by instance, or by individual sample.

This feature also allows you to configure a FlowController for multiple *DataWriters* or a Multi-channel *DataWriter*, so that samples from prioritized writer queues will have precedence when transmission resources become available.

Please refer to the *User's Manual* or HTML documentation for details on the following related changes:

- ❑ New <*type*>DataWriter operations:
  - **write_w_params()**
  - **unregister_instance_w_params()**
  - **dispose_w_params()**
- ❑ Changes to QoS policies:
  - The PublishModeQosPolicy has a new member, **priority**.
  - The MultiChannelQosPolicy has a new member, **priority**, within the DDS_ChannelSettings_t structure.
- ❑ New Value for **scheduling_policy** in FlowControllerProperty_t:
  - The **scheduling_policy** member in FlowControllerProperty_t can be set to a new value, DDS_HPF_FLOW_CONTROLLER_SCHED_POLICY (Highest Priority First). When using this value, the FlowController determines the next destination queue to service based on the publication priorities of the *DataWriter*, a channel of a Multi-channel *DataWriter*, and the individual sample.
- ❑ New Priority-Based Content Filter
  - There is a new built-in content filter named DDS_PRIFILTER_NAME that can filter on data priority.

3 Version Numbers Provided in Header Files

The version numbers for the release are now defined in macros in the header file, **ndds/ndds_c.h**:

```
#define RTI_DDS_VERSION_MAJOR 4
#define RTI_DDS_VERSION_MINOR 5
#define RTI_DDS_VERSION_RELEASE 'e'
#define RTI_DDS_VERSION_REVISION 00
```

These macros make it easy to write code that depends on the *RTI Data Distribution Service* version. For example:

```
#include <ndds/ndds_c.h> // defines the macros (ndds_cpp.h does too)
void MyPublisher() {
    //
    // ...
    //
   #if (RTI_DDS_VERSION_MAJOR == 4 && RTI_DDS_VERSION_MINOR >= 5) ||
       RTI_DDS_VERSION_MAJOR > 4

   // This QoS was introduced in 4.5.
   // When building against 4.4 or below, compilation will
   // still work, but this code will be ignored.
   //
   dataWriterQos.writer_resource_limits.instance_replacement =
                         DDS_UNREGISTERED_INSTANCE_REPLACEMENT;
   #endif
    // Common code for all versions
    dataWriterQos.reliability.kind = DDS_RELIABLE_RELIABILITY;
    DDS_Publisher_create_datawriter(publisher, &dataWriterQos, ...);
    //
    // ...
    //
}
```

[RTI RFE # 248]

4 UDPv6 Transport Support on Select Platforms

The UDPv6 transport is now supported on the VxWorks 6.7-6.9 platforms, and all Linux and Fedora platforms *except* those listed below.

UDPv6 transport is *not* supported on:

- ❏ Freescale™ P2020RDB kernel v2.6.32 (ppc85xxLinux2.6gcc4.3.2)
- ❏ Red Hat® 8 (i86Linux2.4gcc3.2.2)
- ❏ Red Hat 9 (i86Linux2.4gcc3.2)
- ❏ SELinux kernel v2.6.32 (ppc4xxFPLinux2.6gcc4.5.1)
- ❏ Wind River® Linux 3 (ppc85xxWRLinux2.6gcc4.3.2)

Platforms that do not support UDPv6 will ignore the configuration.

5 Simplified Mapping of Topic Names to Multicast Addresses

This release adds support for the new TransportMulticastMappingQosPolicy, which simplifies configuration and assignment of *DataReaders*' multicast addresses. It specifies the automatic mapping between a list of topic expressions and multicast addresses that can be used by a *DataReader* to receive data for a specific topic.

When using this QosPolicy, the middleware will automatically assign a multicast receive address for a *DataReader* from an address range by using configurable mapping rules. The middleware provides a default hash (md5) mapping function, or you can specify a custom mapping function[1].

In addition, the TransportMulticastQosPolicy has a new member, **kind**, which can be set so that the multicast address is selected automatically or set to a ‘unicast-only mode.’ When using the TransportMulticastMappingQosPolicy, this new **kind** must be set to AUTOMATIC.

Please refer to the *User’s Manual* or HTML documentation for more details.

This feature is only available when using the C or C++ API on a platform that supports multicast on a UDPv4 or UDPv6 transport.

1. Custom mapping function currently not supported on VxWorks or VxWorks MILS platforms.

6 Ability to Remove Entries from Peers List

The *DomainParticipant* class has a new API: **remove_peer()**. As the name suggests, it can be used to remove a specific value from the peers list.

7 Ability to Configure Reliable Reader's Send-Window Size

The DataReaderProtocolQosPolicy has a new field, **rtps_reliable_reader.receive_window_size**, which configures the number of out-of-order samples that a *DataReader* can keep.

A larger send-window allows more out-of-order samples to be kept. When the window is full, any subsequent out-of-samples received will be dropped. Such drops would necessitate NACK repairs that would degrade throughput. So, in network environments where out-of-order samples are more probable or where NACK repairs are costly, this window likely should be increased. By default, the window is set to 256, which is the maximum number of samples a single NACK submessage can request.

8 New RTPS Pure NACK Submessage

This release implements a new pure NACK RTPS submessage, which enables *DataReaders* to request more than 256 samples per repair session.

9 Ability to Configure DataReader's Round-trip Time

The DataReaderProtocolQosPolicy has a new field, **rtps_reliable_reader.round_trip_time**, which specifies the estimated round-trip time for messages between a reliable *DataReader* and its matching reliable *DataWriters*. This value is used to ensure that no redundant NACKs will be sent within a single round-trip-time.

10 Multicast-Heartbeat Improvements

Periodic heartbeats can now be sent using the *DataReader's* multicast destination when available. If multicast is not available, periodic heartbeats are sent over unicast. This also applies to late-joiner periodic heartbeats.

Related New Fields in DDS_RtpsReliableWriterProtocol_t:

- **enable_multicast_periodic_heartbeat**

  This boolean value controls whether or not periodic heartbeat messages are sent over multicast.

  When enabled, if a *DataReader* has a multicast destination, the *DataWriter* will send its periodic HEARTBEAT messages to that destination.

  Otherwise, if not enabled or the *DataReader* does not have a multicast destination, the *DataWriter* will send its periodic HEARTBEATs over unicast.

- **multicast_resend_threshold**

  Sets the minimum number of requesting *DataReaders* needed to trigger a multicast resend.

11 Ability to Configure Discovery Plugins in XML

This release adds the ability to load and configure Discovery plugins, such as *RTI Limited Bandwidth Plugins*, using DDS XML QoS profile files, similar to how transport plugins are loaded. In the previous releases, these types of plugins had to be created and attached programatically. See the *RTI Limited Bandwidth Plugins User's Manual* to learn how to configure a *DomainParticipant's* QoS to load these discovery plugins.

12 Cleanup of Discovery and Historical Data

A new feature offers a way to clean up local memory and bandwidth usage (from retransmissions to late-joining *DataReaders*) when an instance is no longer needed.

This feature is enabled through a new field in the WriterDataLifecycleQosPolicy:

- **autopurge_unregistered_instance_delay**

When this feature is enabled, the middleware will clean up all the resources associated with an unregistered instance (most notably, the sample history of non-volatile *DataWriters*) when all the instance's samples have been acknowledged by all its live *DataReaders*, including the sample that indicates the unregistration.

By default, **autopurge_unregistered_instance_delay** is disabled (the delay is INFINITE). If the delay is set to zero, the *DataWriter* will clean up as soon as all the samples are acknowledged after the call to **unregister()**. A non-zero value can be useful in two ways:

1. To keep the historical samples for late-joiners for a period of time.
2. In the context of discovery, if the applications temporarily lose the connection before the unregistration (which represents the remote entity destruction), to provide the samples that indicate the dispose and unregister actions once the connection is reestablished.

This option is also available for discovery data through these new fields in the DiscoveryConfigQosPolicy:

❑ **publication_writer_data_lifecycle.autopurge_unregistered_instances_delay**

❑ **subscription_writer_data_lifecycle.autopurge_unregistered_instances_delay**

Note: This feature is not currently supported in the ODBC writer-history plug-in; it works in the (default) in-memory writer history only.

[RTI RFE # 358 and 439]

13 Ability to Configure Custom FlowControllers in XML

FlowControllers can now be created and configured using the *DomainParticipant's* Property QoS policy. The properties can be specified in an XML profile, or programmatically.

This also enables other RTI products, such as *RTI Routing Service* and *RTI Persistence Service*, to use custom FlowControllers.

14 Ability to Configure Logging in XML

This release adds the ability to configure logging via XML using a new LoggingQosPolicy, which is part of DomainParticipantFactoryQos. This allows you to specify the logging verbosity, category, print format, and output file. See the *User's Manual* (Sections 8.4.1 and 18.2.2) for more details.

15 Support for Non-UP Interfaces for UDPv4 Transport

This release adds the ability to consider interfaces which were down at the time of participant creation so that whenever those interfaces become available the application will be able to use them.

In this release, this feature is only supported on Windows platforms with statically configured IP addresses.

To support this new feature, there are two new UDPv4 transport properties, **ignore_nonup_interfaces** and **interface_poll_period**.

You can use **ignore_nonup_interfaces** to allow/disallow the use of interfaces that are not reported as UP (by the operating system) in the UDPv4 transport.

Please see the *RTI Data Distribution Service User's Manual* for more information on these new properties (Table 13.2).

16 TransportPriority QoS Support for UDPv6 Added to Linux 2.6 Platforms

The TransportPriorityQosPolicy for the UDPv6 transport is now supported on Linux 2.6 platforms (in addition to Solaris 2.10 platforms).

[RTI Bug # 13872]

17 New UDPv4 Transport Property

The new UDPv4 Transport property, **reuse_multicast_receive_resource**, controls whether or not to reuse receive resources. Setting this to 0 (FALSE) prevents multicast crosstalk by uniquely configuring a port and creating a receive thread for each multicast group address. This property affects Linux systems only.

18 New APIs to Read/Take Instance with ReadCondition

Two new APIs are available to read or take instances: **read_instance_w_condition()** and **take_instance_w_condition()**. They are similar to **read_instance()** and **take_instance()**, but all returned samples belong to the single specified instance *and* satisfy a specified ReadCondition. For details, see the online documentation or Section 7.4.3.7 in the *User's Manual*.

See the updated *Platform Notes* document for more information.

19 Additional Formatting Technique for Participant IDs

When formatting a peer descriptor, you can now use brackets around the participant ID to indicate that you only want to use a specific integer (such as [6]) or range of integers (such as [4,7]).

20 Documentation for Reserved KeyWords

The *User's Manual* (Section 3.3) now includes a list of reserved IDL keywords.

21 New TCP Transport Property to Limit Blocking Time

A new TCP transport property, **send_max_wait_sec**, controls the maximum time (in seconds) the low-level **sendto()** function is allowed to block the caller thread when the

TCP send buffer becomes full. Limiting this delay eliminates the possibility of deadlock and increases the response time of the internal DDS thread.

[RTI Bug # 14053]

22 New Transport Plugin API for C and C++

The new **get_transport_plugin()** operation retrieves a transport plug-in registered in a *DomainParticipant* by its alias.

This operation is only available in the C and C++ APIs.

23 New APIs to Get All DataWriters and DataReaders

The DDS_Publisher class has a new API to retrieve all the *DataWriters* for a given *Publisher*, **get_all_datawriters().** There is a parallel API for *Subscribers* to get all *DataReaders*, **get_all_datareaders()**.

24 New Architectures

This release adds support for the following architectures:

Operating System		CPU	Compiler	RTI Architecture Abbreviation
INTEGRITY	INTEGRITY 10.0.0	Pentium	CHnet IPv4 stack	pentiumInty10.0.0.pcx86
Linux (Cell BE)	Fedora 12 (2.6.32 kernel) with gcc 4.5.1	64-bit Cell	gcc 4.5.1	cell64Linux2.6gcc4.5.1

Operating System		CPU	Compiler	RTI Architecture Abbreviation
Linux (Intel)	Fedora 12 (2.6.32 kernel)	x64	gcc 4.4.4	x64Linux2.6gcc4.4.4
	Fedora 12 (2.6.32 kernel) with gcc 4.5.1	x64	gcc 4.5.1	x64Linux2.6gcc4.5.1
	Red Hat Enterprise Linux 6.0 and 6.1 (2.6 kernel)	Pentium	gcc 4.4.5	i86Linux2.6gcc4.4.5
			Sun Java Platform Standard Edition JDK 1.6	i86Linux2.6gcc4.4.5jdk
		x64	gcc 4.4.5	x64Linux2.6gcc4.4.5
			Sun Java Platform Standard Edition JDK 1.6	x64Linux2.6gcc4.4.5jdk
Linux (PowerPC)	Freescale P2020RDB (2.6.32 kernel)	PPC 85xx	Freescale gcc.4.3.74 based on gcc.4.3.2	ppc85xxLinux2.6gcc4.3.2
	SELinux (2.6.32 kernel)	PPC 4xxFP	gcc 4.5.1	ppc4xxFPLinux2.6gcc4.5.1
	Wind River Linux 3	PPC 85xx	gcc 4.3.2	ppc85xxWRLinux2.6gcc4.3.2
QNX	QNX Neutrino® 6.4.1	Pentium	qcc 4.3.3 with GNU C++ libraries	i86QNX6.4.1qcc_gpp
	QNX Neutrino 6.5	Pentium	qcc 4.4.2 with GNU C++ libraries	i86QNX6.5qcc_gpp4.4.2

Operating System		CPU	Compiler	RTI Architecture Abbreviation
VxWorks	VxWorks 6.8	Pentium	gcc 4.1.2	For Kernel Modules: pentiumVx6.8gcc4.1.2 For Real Time Processes: pentiumVx6.8gcc4.1.2_rtp
		Any Wind River PPC32 CPU with floating point hardware	gcc 4.1.2	For Kernel Modules: ppc604Vx6.8gcc4.1.2 For Real Time Processes on a non-SMP system: ppc604Vx6.8gcc4.1.2_rtp
	VxWorks 6.9	Pentium32-bit	gcc 4.3.3	For Kernel Modules: pentiumVx6.9gcc4.3.3 For Real Time Processes: pentiumVx6.9gcc4.3.3_rtp
		Pentium 64-bit	gcc 4.3.3	For Kernel Modules: pentium64Vx6.9gcc4.3.3 For Real Time Processes: pentium64Vx6.9gcc4.3.3_rtp
		Any Wind River PPC32 CPU with floating point hardware	gcc 4.3.3	For Kernel Modules: ppc604Vx6.9gcc4.3.3 For Real Time Processes: ppc604Vx6.9gcc4.3.3_rtp
VxWorks 653	VxWorks 653 2.3	sbc8641d	gcc 3.32	sbc8641Vx653-2.3gcc3.3.2
		SIMPC	gcc 3.32	simpcVx653-2.3gcc3.3.2
VxWorks MILS	VxWorks MILS 2.1.1 with vThreads 2.2.2 (VxWorks 5.5.1) Guest OS	PPC604	gcc 3.3.2	ppc604VxT2.2.2gcc3.3.2