RTI Routing Service

Release Notes

Version 5.2.3

Your systems. Working as one.

Printed in U.S.A. First printing.
April 2016.

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

Release Notes

1 Supported Platforms

RTI® Routing Service is supported on the platforms in Table 1.1. It can also be deployed as a C library linked into your application. This is true for all platforms in Table 1.1 except INTEGRITY.

Table 1.1 **Supported Platforms**

Platform	Operating System	Architecture
INTEGRITY® [1,2]	INTEGRITY 10.0.2	pentiumInty10.0.2.pcx8
	INTEGRITY 11.0.4	pentiumInty11.pcx86-smp
Linux®	CentOS® 5.4, 5.5	i86Linux2.6gcc4.1.2 x64Linux2.6gcc4.1.2
	CentOS 6.0, 6.2 - 6.4	i86Linux2.6gcc4.4.5 x64Linux2.6gcc4.4.5
	CentOS 7.0	i86Linux3.xgcc4.8.2 x64Linux3.xgcc4.8.2
	Raspbian Wheezy 7.0	armv6vfphLinux3.xgcc4.7.2
	Red Hat® Enterprise Linux 5.0	i86Linux2.6gcc4.1.1 x64Linux2.6gcc4.1.1
	Red Hat Enterprise Linux 5.1, 5.2, 5.4, 5.5	i86Linux2.6gcc4.1.2 x64Linux2.6gcc4.1.2
	Red Hat Enterprise Linux 6.0 - 6.5, 6.7	i86Linux2.6gcc4.4.5 x64Linux2.6gcc4.4.5
	Red Hat Enterprise Linux 7	i86Linux3.xgcc4.8.2 x64Linux3.xgcc4.8.2
	Ubuntu® Server 12.04 LTS	i86Linux3.xgcc4.6.3 x64Linux3.xgcc4.6.3
	Ubuntu Server 14.04 LTS	i86Linux3.xgcc4.8.2 x64Linux3.xgcc4.8.2
Mac®	All Mac OS X platforms listed in the *RTI Connext DDS Core Libraries Platform Notes* for the same version number	
Windows®	All Windows platforms listed in the *RTI Connext DDS Core Libraries Platform Notes* for the same version number	

1. Supported both as an executable and as a C or Java library, unless stated otherwise.

2. Does not include TCP/IPv4 transport plugin; implemented as a static library.

Routing Service is also supported on the platforms listed in Table 1.2; these are target platforms for which RTI offers custom support. If you are interested in these platforms, please contact your local RTI representative or email **sales@rti.com**.

Table 1.2 **Custom Supported Platforms**

Operating System	CPU	Compiler	RTI Architecture Abbreviation
NI Linux Real-Time 3.2[1]	ARMv7	gcc 4.4.1	armv7AngstromLinux3.2gcc4.4.1.cortex-a9

1. Requires NI-RIO 13.1 release or a patch from NI for NI-RIO 13.0

2 Supported Languages

- ❏ The transformation plugin API is only available for C.
- ❏ The adapter plugin API is available for C and Java.

3 Compatibility

3.1 Connext DDS Compatibility

Routing Service can be used to forward and transform data between applications built with *RTI Connext DDS*™, as well as *RTI Data Distribution Service* 4.5[b-e], 4.4d, 4.3e, and 4.2e except as noted below.

- ❏ *Routing Service* is not compatible with applications built with *RTI Data Distribution Service* 4.5e and earlier releases when communicating over shared memory. For more information, please see the Transport Compatibility section in the *RTI Connext DDS Core Libraries Release Notes*.
- ❏ Starting in *Connext DDS* 5.1.0, the default **message_size_max** for the UDPv4, UDPv6, TCP, Secure WAN, and shared-memory transports changed to provide better out-of-the-box performance. *Routing Service* 5.1.0 also uses the new value for **message_size_max**. Consequently, *Routing Service* 5.1.0 is not out-of-the-box compatible with applications running older versions of *Connext DDS* or *RTI Data Distribution Service*. Please see the *RTI Connext DDS Core Libraries Release Notes* for instructions on how to resolve this compatibility issue with older *Connext DDS* and *RTI Data Distribution Service* applications.
- ❏ The types of the remote administration and monitoring topics in 5.1.0 are not compatible with 5.0.0. Therefore:
  - The 5.0.0 *RTI Routing Service* shell, *RTI Admin Console* 5.0.0, and *RTI Connext DDS* 5.0.0 user applications performing monitoring/administration are not compatible with *RTI Routing Service* 5.1.0.
  - The 5.1.0 *RTI Routing Service* shell, *RTI Admin Console* 5.1.0, and *RTI Connext DDS* 5.1.0 user applications performing monitoring/administration are not compatible with *RTI Routing Service* 5.0.0.

Routing Service works out-of-the box with unbounded types due to changes in the default values for memory management (see Changed Default Values for Memory Management in Participant Configuration (Section 6.1.7)). These values are not set to a finite values, so that memory is not

allocated to the maximum (which, for unbounded types, would likely cause a memory overflow).

3.1.1 RTI Data Distribution Service 4.2e Compatibility

If the applications' data types contain 8-byte or larger primitive types (double, long long, unsigned long long or long double), *Routing Service* will have to be run with the command line option **-use42eAlignment** in order to be compatible with *RTI Data Distribution Service* 4.2e.

If the applications use large data, *Routing Service* must be configured with the following properties set to 1 in order to be compatible with *RTI Data Distribution Service* 4.2e:

- **dds.data_writer.protocol.use_43_large_data_format**
- **dds.data_reader.protocol.use_43_large_data_format**

3.1.2 RTI Data Distribution Service 4.3e Compatibility

If the applications use large data, *Routing Service* must be configured with the following properties set to 1 in order to be compatible with *RTI Data Distribution Service* 4.3e.

- **dds.data_writer.protocol.use_43_large_data_format**
- **dds.data_reader.protocol.use_43_large_data_format**

3.2 Command-Line Options Compatibility

Starting with *Routing Service* 1.1.0, the command-line parameter **-srvName** has been replaced with **-cfgName** (to select a configuration) and **-appName** (to name the service execution). In previous *Routing Service* versions, the **-srvName** parameter was not required. However, in this version the equivalent parameter **-cfgName** is required.

Also starting with *Routing Service* 1.1.0, the allowed values for the **-verbosity** command-line option changed. The new **-verbosity** option coalesces the old **-verbosity** and **-ddsVerbosity** options into a single parameter.

For additional information about command-line options, see Chapter 3 in the *Routing Service Getting Started Guide*.

3.3 XML Compatibility

- Starting with *Routing Service* 1.1.0, the attribute "name" in the **<routing_service>** tag is now required. Old XML files without the name attribute will not be parsed by *Routing Service* 1.1.0 and higher.
- Starting with *Routing Service* 2.0.0, the way to register and configure transformations in the configuration file has changed:
  - The tag <transformation_class_library> has been replaced with <transformation_library>
  - The tag <transformation_class> has been replaced with <transformation_plugin>.
  - The content of these tags has also changed. With the new configuration there is only a single entry point to the library. For example:

```
<transformation_library name="MyTransfLib">
      <transformation_plugin name="MyTransfPlugin">
            <dll>mytransformation</dll>
            <create_function> <!-- Entry point -->
                  MyTransfPlugin_create
            </create_function>
      </transformation_plugin>
</transformation_library>
```

- The configuration of a transformation within <route> is done using properties instead of the <expression> and <parameter> tags. For example:

```
<transformation plugin_name="TransformationLib::Assignment">
      <property>
            <value>
                  <element>
                        <name>X</name>
                        <value>Y</value>
                  </element>
                  <element>
                        <name>Y</name>
                        <value>X</value>
                  </element>
            </value>
      </property>
</transformation>
```

- The configuration of the assignment transformation distributed with *Routing Service* is now done with properties. For example:

  Before 2.0.0:

```
<transformation className="TransformationLib::Assignment">
      <expression></expression>
      <parameter>position.x=position.y</parameter>
      <parameter>x=10</parameter>
</transformation>
```

  In 2.0.0 and higher:

```
<transformation plugin_name="TransformationLib::Assignment">
      <property>
            <value>
                  <element>
                        <name>position.x</name>
                        <value>position.y</name>
                  </element>
                  <element>
                        <name>x</name>
                        <value>10</name>
                  </element>
            </value>
      </property>
</transformation>
```

3.4 Transformation API

The transformation API of *Routing Service* 2.0.1 and higher is not compatible with the API of previous releases (2.0.0 and lower).

The new API follows the same model as the adapter API which introduced the concept of a Plugin as a C structure that contains all the function pointers that implement the interface.

The registration of a transformation plugin with the new model requires a single entry-point to the shared library; the entry-point is a function that creates the Plugin structure which contains the implementation.

For example:

```
<transformation_library name="MyTransfLib">
```

```
            <transformation_plugin name="MyTransfPlugin">
                    <dll>mytransformation</dll>
                    <create_function> <!-- Entry point -->
                            MyTransfPlugin_create
                    </create_function>
            </transformation_plugin>
    </transformation_library>
```

The following table shows how deprecated functions map to the new API.

2.0.0 API (rti_routingservice.h)	2.0.1 and Higher API (routingservice_transformation.h)	Comments
RTITransformationClass_ loadFnc()	RTI_RoutingServiceTransformation Plugin_CreateFcn()	This is the entry-point function.
RTITransformationClass_ unloadFnc()	Function declaration: RTI_RoutingServiceTransformation Plugin_DeleteFcn() Member in Plugin struct: transformation_plugin_delete	
RTITransformationClass_ createFnc()	Function declaration: RTI_RoutingServiceTransformation Plugin_CreateTransformationFcn() Member in Plugin struct: transformation_plugin_create_ transformation	
RTITransformationClass_ deleteFnc()	Function declaration: RTI_RoutingServiceTransformation Plugin_DeleteTransformationFcn() Member in Plugin struct: transformation_plugin_delete_ transformation	
RTITransformationClass_ modifyFnc()	Function declaration: RTI_RoutingServiceTransformation_ UpdateFcn() Member in Plugin struct: transformation_update	
RTITransformationClass_ transformFnc()	Function declaration: RTI_RoutingServiceTransformation_ TransformFcn() Member in Plugin struct: transformation_transform	In the new API, the transform function accept multiple samples. In addition, the output samples must be created by the transformation instead of being passed in by *Routing Service*.
(none)	Function declaration: RTI_RoutingServiceTransformation_ ReturnLoanFcn() Member in Plugin structure: transformation_return_loan	This function is used to return the loan on the samples returned by the transform function.

4 What's New in 5.2.3

This release adds support for these new platforms:

- ❏ Cent OS 7.0
- ❏ Mac OS X 10.11
- ❏ Red Hat Enterprise Linux 6.7

For details on these platforms, see the *RTI Connext DDS Core Libraries Platform Notes.*

5 What's Fixed in 5.2.3

5.1 Connext DDS only Printed Exception Messages, Ignored Verbosity Settings

Verbosity settings in *Routing Service* did not affect what type of messages were printed by *Connext DDS*. Only DDS exception messages were printed, regardless of the verbosity setting. This issue, which was a side effect of CORE-6075, has been resolved.

[RTI Issue ID ROUTING-261]

5.2 Routing Service Crashed at Loading Time if Invalid Participant References in Configuration

Routing Service would crash if the configuration contained AutoTopicRoute/TopicRoute definitions whose input or output referenced a non-participant connection. The crash would occur when *Routing Service* tried to load the configuration. Before the crash, the following message was printed in the output:

```
ROUTERCfgFileParticipant_narrow: Element "::RouterService1::adapter2dds::1"
is not a XML participant object
```

This problem has been resolved. Now in this situation, *Routing Service* will fail and the process will exit without crashing.

[RTI Issue ID ROUTING-344]

5.3 Race Condition in DDS Adapter may have Caused Routing Service to Crash

A race condition while accessing discovery information within the DDS adapter may have caused *Routing Service* to crash if an endpoint was discovered or disposed. This problem has been resolved.

[RTI Issue ID ROUTING-350]

5.4 Remote Creation of DomainRoute could Fail

The remote creation of DomainRoute may have failed if the associated configuration would contain one or more definitions of Route/TopicRoute and/or AutoRoute/AutoTopicRoute. The output would have shown the following log:

```
ROUTERCfgFileStreamPort_initialize:Parse error at line 52: get referenced
connection configuration
ROUTERCfgFileAutoRouteStreamPort_initialize:!init ROUTERCfgFileStreamPort
object
ROUTERCfgFileStreamPortFactory_new:!init ROUTERCfgFileAutoRouteStreamPort
object
```

```
RTIXMLParser_onStartTag:Parse error at line 52: Error processing tag
'input'
RTIXMLParser_parseFromString:error parsing XML string
DDS_XMLParser_parse_from_string:Error parsing string
ROUTERCfgFileParser_parseXmlString:error parsing configuration file 'xml
string'
ROUTERRemoteService_createEntity: error parsing XML snippet for creation
RTI_RoutingService_create_entity:!createEntity
RTI_RoutingService_create_entity error
```

This problem has been resolved.

[RTI Issue ID ROUTING-351]

5.5 Unbounded Memory Growth due to Discovery of Endpoints if Remote Administration Enabled

If remote administration was enabled, *Routing Service* used an unbounded amount of memory. This issue was provoked by the built-in reader of the remote administration participant; the information generated by the built-in reader was never cleaned up. This problem has been resolved.

[RTI Issue ID ROUTING-361]

5.6 Routing Service didn't Route Data nor Type Information in Routes Whose Input and Output had Different Registered Type Names

If routes or topic routes were configured such that their input and output routes did not match the registered type name for the route, they would not route data from the input to the output. Likewise, if the <route_types> tag was present and enabled, the type information would not be applied from the input to the output, and vice versa.

This limitation has been removed so the input and output do not have to match the registered type name for the route to operate.

Note: This problem only occurred in 5.2.0.

[RTI Issue ID ROUTING-362]

5.7 TopicRoute/AutoTopicRoute Ignored Registered Type Name for Type Information Retrieved from Discovery and Opposite Connection

If *Routing Service* was configured to retrieve the type information from discovery, the registered type names would be ignored if the type was provided by the opposite connection. TopicRoutes that set <route_types> to true and AutoTopicRoutes could have been created with the wrong type information, which would cause communication to fail. This issue was introduced as part of the fix for ROUTING-362 (see Section 5.6) and has been resolved.

[RTI Issue ID ROUTING-366]

5.8 Race Condition in DDS Adapter could Cause Inconsistent State or Crash

The DDS adapter was sharing some resources that were concurrently accessed unsafely. This caused *Routing Service* to have an inconsistent state or even crash. This situation was more likely to happen during service start up. It could also happen while the service was running, causing unexpected behavior. This problem has been resolved.

[RTI Issue ID ROUTING-370]

5.9 Failure to Write NOT_ALIVE_NO_WRITERS Samples

An output DDS StreamWriter may have generated the following write error when trying to publish a NOT_ALIVE_NO_WRITERS sample propagated by *Routing Service* from an input DDS StreamReader through a Topic Route or Auto-Topic Route:

```
PRESPsWriter_writeInternal:!sequence number order
ROUTERDdsStreamWriter_write:!write (retcode: 3)
ROUTERStreamWriter_write:adapter 'rtidds' error (0): !write (retcode: 3)
```

This problem only occurred when the *DataReader* associated with the input DDS StreamReader lost the matched status with all the user *DataWriters* and the XML tag <propagate_unregister> was set to true in the Topic Route our Auto-Topic Route.

[RTI Issue ID ROUTING-381]

5.10 Service Crashed after Enabling AutoTopicRoute with Monitoring Enabled

If *Routing Service* enabled monitoring and received a remote command to enable an existing disabled AutoTopicRoute, the service may have crashed. The crash occurred during the enable process if the AutoTopicRoute triggered the creation of a TopicRoute and attempted to publish monitoring data. This problem has been resolved.

[RTI Issue ID ROUTING-385]

6 Previous Releases

This section includes:

6.1 What's New in 5.2.0

6.1.1 New Platforms

This release added support for these platforms:

- ❏ INTEGRITY 11.0.4
- ❏ Mac OS X 10.8 and 14
- ❏ Red Hat Enterprise Linux 6.5 and 7
- ❏ Ubuntu 14
- ❏ Windows 8, 8.1; Windows Server 2012 R2

For details on these platforms, see Table 1.1.

6.1.2 Removed Platforms

Windows platforms using Visual Studio 2005 are no longer supported.

6.1.3 Support for External Hardware Load Balancers in TCP Transport Plugin

This release adds support for a new property in NDDS_Transport_TCPv4_Property_t, **negotiate_session_id**. By default, this property is set to FALSE. When set to TRUE, the TCP Transport Plugin will perform a session negotiation that will help external load balancers identify all the connections associated with a particular session between two *Connext* applications.

This keeps the connections from being divided among multiple servers and ensures proper communication. For more information, see the *Routing Service User's Manual* (Section 7.2.4, *Support for External Hardware Load Balancers in TCP Transport Plugin*).

6.1.4 Ability to Link Routing Service as a Library

On all supported architectures (see Table 1.1), *Routing Service* can now be deployed as a C library linked into your application. See the *Routing Service Getting Started Guide* (Section 3.3) or *User's Manual* (Section 3.3) for details.

6.1.5 New Administration APIs

There are new administration APIs to create and delete entities and get the current configuration. See the *Routing Service* API Reference HTML documentation for details.

6.1.6 Java Library for Windows Platform Links Against Visual Studio 2010 Native Libraries

The Java library for use on Windows platforms, i86Win32jdk, now links against the Visual Studio 2010 native libraries.

6.1.7 Changed Default Values for Memory Management in Participant Configuration

Default value for memory management of the participant configuration in a DomainRoute has been modified to be equivalent as the following XML:

```
<memory_management>
    <sample_buffer_min_size>65536</sample_buffer_min_size
    <sample_buffer_trim_to_size>false<sample_buffer_trim_to_size>
</memory_management>
```

6.2 What's Fixed in 5.2.0

6.2.1 Heap Corruption when Using RTI Routing Service

RTI Routing Service may have crashed due to a heap corruption in the *RTI Connext DDS* core libraries. This crash may have occurred anytime after *RTI Routing Service* started routing data for which the data type had FINAL or EXTENSIBLE extensibility. This problem has been resolved.

[RTI Issue ID CORE-6224]

6.2.2 Failure to Route Samples with Size Greater than 65536 Bytes

Routing Service may have failed to route samples whose size was greater than 65536 bytes. When this occurred, the service reported the following error:

```
DDS_DynamicData_copy:unable to grow buffer
```

This problem has been resolved.

[RTI Issue ID ROUTING-219]

6.2.3 Filter Propagation may not have Propagated Stop-Band Filter on StreamWriter Deletion

On an event of StreamWriter deletion, filter propagation may have missed the propagation of a stop-band filter (or its union with the configuration filter if it was present). Instead, the latest filter update would have prevailed with no more updates. This problem has been resolved.

[RTI Issue ID ROUTING-228]

6.2.4 Filter Propagation did not Keep Track of Filters while being Disabled via Remote Administration

If filter propagation was disabled via remote administration, further changes in the filter information from the subscription side were not tracked. Hence, when it was enabled again, it would not propagate the latest state until a new filter event occurred. This problem has been resolved so that right after filter propagation is re-enabled, a filter is propagated reflecting the most up-to-date status of filters from the subscription side.

[RTI Issue ID ROUTING-232]

6.2.5 Service Crashed if Create or Delete Command Received while Service was Stopped

If *Routing Service* was running but was in a stopped state when it received a remote create or delete command, the service would crash.

This problem has been resolved. Now if a remote create or delete command is received while *Routing Service* is stopped, the XML configuration currently in use will be updated to reflect the new or deleted entity. Then the requested creation or deletion will take place when the service resumes.

[RTI Issue ID ROUTING-264]

6.2.6 Removed Participant Name from Any Participant QoS in USER_ROUTING_SERVICE.xml

The participant name that was explicitly set in the participant QoS definition in USER_ROUTING_SERVICE.xml has been removed. This was done to avoid issues with Admin Console, which would check on this name in order to determine whether a participant belonged to a specific service.

[RTI Issue ID ROUTING-277]

6.2.7 No Error Reported when Remote Domain-Route Creation Failed, could Result in Inconsistent State and Possible Crash

Routing Service did not report an error to the Remote Administration client when a **create** command failed to create a domain route. This may have resulted in an inconsistent state and *Routing Service* may have crashed. This problem has been resolved. Now the client will receive an error and *Routing Service* will log it locally.

[RTI Issue ID ROUTING-280]

6.2.8 Problems with Filter Propagation Depending on Order of Subscription Discovery

If *Routing Service* discovered user DataReaders in a certain order, filter information was not updated properly. This caused incorrect propagation of the composed filter. This problem has been resolved; now the order of discovery does not affect filter propagation.

[RTI Issue ID ROUTING-287]

6.2.9 Unbounded Memory Growth Caused by Internal Discovery State not being Deleted

Routing Service created internal discovery state that was not deleted until *Routing Service* shut down, even when such state was no longer needed. This caused *Routing Service* to have unlimited memory growth over time. This problem has been resolved.

[RTI Issue ID ROUTING-308]

6.2.10 Unbounded Memory Growth Caused by Remote Creation of Entities

If an entity was created via remote administration, certain memory associated with the related XML object was not freed until shutdown. This problem has been resolved.

[RTI Issue ID ROUTING-311]

6.2.11 Unexpected "DDS_DynamicData_from_key_stream:deserialization error: reserialize member" Error Message

You may have seen the following error message:

```
DDS_DynamicData_from_key_stream:deserialization error: reserialize member
```

This problem only occurred when the tag <filter_propagation> in a route was set to 1 and did not affect correctness. This problem has been resolved.

[RTI Issue ID ROUTING-314]

6.2.12 Sending Load Command Disabled Distributed Logger Even if it was Enabled in New Configuration

When a **load** command is sent to *Routing Service*, the Distributed Logger's instance was destroyed and did not come back even if Distributed Logger was enabled in the new configuration. As a result, tools such as Admin Console could not visualize log messages coming from the *Routing Service* instance. This problem has been resolved.

[RTI Issue ID ROUTING-320]

6.2.13 Memory Corruption when Sending GET or SAVE Commands to Routing Service

Routing Service may have crashed if a **get** or **save** command was received and the *Routing Service* configuration contained a QoS profile with a **writer_qos** or **reader_qos** that inherited from a **topic_qos**. For example:

```
<qos_library name="lib1">
    <topic_qos name"topic_qos">
    </topic_qos>
</qos_library>

<qos_library name="lib2">
    <writer_qos base_name"=lib1::topic_qos">
     ....
    </writer_qos>
<qos_library>
```

This problem has been resolved.

[RTI Issue ID ROUTING-341]

7 Known Issues

7.1 Sequences of Transformations in a Route not Supported

The tag <transformation_sequence> within a <topic_route> is not supported. Only one transformation per route is supported.

[RTI Issue ID ROUTING-4]

7.2 Assignment Data Transformation only Supports Assignment of Primitive Fields not Part of Arrays or Sequences

The data transformation library distributed with *Routing Service* only supports the assignment of primitive fields (including strings) that are not part of arrays or sequences.

For example:

```
<transformation className="TestTransformationLib::FieldMapping">
  <expression></expression>
  <parameter>position.x=position.y</parameter> <!- This is supported ->
  <parameter>x=y</parameter>                  <!- This is supported ->
  <parameter>x[0]=y[0]</parameters>            <!- This is not supported ->
  <parameter>position=position</parameter>     <!- This is not supported ->
</transformation>
```

For additional details about data transformation, see Chapter 3 in the *Routing Service User's Manual*.

[RTI Issue ID ROUTING-7]

7.3 Limitations in Adapter API

In the Adapter API, **Connection::get_attributes()** and update operations are currently not supported.

[RTI Issue ID ROUTING-222]

7.4 Topics of Data Types with Bit Fields are not Supported

Routing Service cannot communicate with DataReaders or DataWriters of Topics with a data type that includes bit fields. You may see the following messages, but *Routing Service* will continue to work normally otherwise:

```
DDS_DynamicDataTypeSupport_initialize:type not supported (bitfield member)
ROUTERTypeInfo_initialize:!create dynamic data type support
ROUTERTypeInfo_new:!init ROUTERTypeInfo object
ROUTERDdsConnection_assertType:!create type info
```

[RTI Issue ID CORE-3949]