RTI Routing Service

Release Notes

Version 5.1.0

Your systems. Working as one.

Printed in U.S.A. First printing.
December 2013.

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 these platforms:

Table 1.0 **Supported Platforms**

Platform	Operating System	Architecture
INTEGRITY®	INTEGRITY 10.0.2	pentiumInty10.0.2.pcx86[1]
Linux®	CentOS 5.4, 5.5 (2.6 kernel)	i86Linux2.6gcc4.1.2 x64Linux2.6gcc4.1.2
	CentOS 6.0, 6.2 - 6.4	i86Linux2.6gcc4.4.5 x64Linux2.6gcc4.4.5
	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 - 6.4	i86Linux2.6gcc4.4.5 x64Linux2.6gcc4.4.5
	Ubuntu® Server 12.04 LTS (2.6 kernel)	i86Linux3.xgcc4.6.3 x64Linux3.xgcc4.6.3
Windows®	Windows 7 Windows 8 Windows Server® 2008 R2 Windows Server 2012 R2 Windows 2003 Windows Vista® Windows XP Professional	i86Win32VS2005 i86Win32VS2008 i86Win32VS2010 i86Win32VS2012 x64Win64VS2005 x64Win64VS2008 x64Win64VS2010 x64Win64VS2012

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

Routing Service is also supported on the platforms listed in Table 1.1; 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 1.1 **Custom Supported Platforms**

Operating System	CPU	Compiler	RTI Architecture Abbreviation
NI Linux	ARMv7	gcc 4.4.1	armv7AngstromLinux3.2gcc4.4.1.cortex-a9

2 Supported Languages

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

3 Compatibility

3.1 Connext Compatibility

Routing Service can be used to forward and transform data between applications built with *RTI Connext™*, 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 Core Libraries and Utilities Release Notes*.[1]
- ❏ In *Connext* 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* or *RTI Data Distribution Service*. Please see the *RTI Core Libraries and Utiltities Release Notes* for instructions on how to resolve this compatibility issue with older *Connext* 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* 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* 5.1.0 user applications performing monitoring/administration are not compatible with *RTI Routing Service* 5.0.0.

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:

1. See <*Connext installation directory*>/ndds.<*version*>/doc/pdf/RTI_CoreLibrariesAndUtilities_ReleaseNotes.pdf.

- ❏ **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 5.1.0

4.1 New Platforms

This release adds support for the following platforms:

- ❏ CentOS 6.2 - 6.4
- ❏ Raspbian Wheezy 7.0
- ❏ Red Hat Enterprise Linux 6.2 - 6.4
- ❏ Ubuntu Server 12.04 LTS (2.6 kernel)
- ❏ Windows 8
- ❏ Windows Server 2012 R2

4.2 Filter Propagation

This release enables support for propagation of the filter information from DataReaders using a ContentFilteredTopic. This feature can be enabled by using the tag <filter_propagation> within <topic_route>. See Section 2.4.6 and Chapter 9 in the *RTI Routing Service User's Manual*.

4.3 Ability to Override Name of Participants Created by Routing Service

When *Routing Service* creates a DomainParticipant, it uses a specific format for its name (which gets propagated on the wire):**RTI Routing Service: [*service name*].[*domain route name*]#[1|2]**, where **[*service name*].[*domain route name*]** are the names of the service and domain route that contain that participant, followed by 1 or 2, indicating if it is the first or second participant in that domain route.

In previous releases, it was not possible to change this name. Now, when specified, *Routing Service* will use the name specified through the DomainParticipant QoS.

For example, when using the following configuration, *Routing Service* will create a participant named **MyParticipant**:

```
<routing_service name="MyService">
  <domain_route name="MyDomainRoute">
    <participant_1>
      ...
      <participant_qos>
         <participant_name><name>MyParticipant</name></participant_name>
      </participant_qos>
    </participant_1>
    ...
  </domain_route>
  ...
</routing_service>
```

If you omit the line, <participant_name><name>MyParticipant</name></participant_name>, *Routing Service* will use the default name as described above: **RTI Routing Service: MyService.MyDomainRoute#1**.

The DomainParticipants created for remote administration and monitoring will also honor user-specified names.

5 What's Fixed in 5.1.0

5.1 Possible Crash During Transformation if Monitoring Enabled

Routing Service may have crashed while gathering statistics for samples that go through a transformation. This problem only occurred if monitoring was enabled and a transformation plug-in released a sample in its **return_loan()** function.

[RTI Issue ID ROUTING-139]

5.2 Possible Segmentation Fault upon Discovery of Types with Bit Fields

If *Routing Service* discovered a DataReader or DataWriter whose type contained bit fields, it may have issued a segmentation fault. *Routing Service* does not support bit fields (see Section 6.4), but now it will handle the error appropriately.

[RTI Issue ID ROUTING-162]

5.3 Possible Crash when Updating Any Routing Service Entity with Invalid Configuration when Disabled

Routing Service may have crashed if an update operation of any entity was performed on a disabled service. If the updating configuration provoked an error, a crash may have occurred while retrieving the remote administration logger used to report the update error. This problem has been resolved.

[RTI Issue ID ROUTING-171]

5.4 Possible Crash after Updating Disabled DomainRoute

Updating a disabled Domain Route, for example to change an immutable DomainParticipant QoS policy, may have caused *Routing Service* to crash. This problem has been resolved.

[RTI Issue ID ROUTING-190]

5.5 Routing Service Entered Infinite Loop upon XML DTD Validation Error During Start Up

If an XML DTD validation error occurred while Routing Service was starting, it may have hung in an infinite loop due to finalization cleanup. This problem has been resolved.

[RTI Issue ID ROUTING-192]

5.6 Deserialization Errors when Using Topic Route if Input Type Contained Unions or Sequence Members

Routing Service may have produced deserialization errors upon sample reception if the service configuration contained a topic route where the input type contained members whose types were sequences or unions. This issue has been resolved.

[RTI Issue ID ROUTING-150]

5.7 Deserialization Errors if Topic Route's Input Type was a Valuetype

Routing Service may have produced deserialization errors like the one below if the service configuration contained a topic route in which the input type was a valuetype.

```
DDS_DynamicData_from_stream:ERROR: Failed to get sample serialized size
```

This issue has been resolved.

[RTI Issue ID ROUTING-153]

5.8 Remote Update Failed when Sending Well-Formed Service Configuration with Default Name

If a well-formed string configuration was sent in a command to update *Routing Service*, and that configuration had the same name as any of the default ones, the operation would fail with a parser error. This issue was noticeable when using *RTI Admin Console* to modify the service configuration when it was loaded with a default one.

This issue has been resolved; now when this situation happens, the updating configuration will override the default one with the same name.

[RTI Issue ID ROUTING-169]

5.9 Incorrect Handling of Samples if Received Type was Valuetype Marked as Extensible

Routing Service did not correctly handle samples if the received type was a valuetype marked as extensible. The following error was reported:

```
DDS_DynamicData_from_stream:ERROR: Failed to get sample serialized size
```

This only happened when the valuetype was marked as extensible and it was the top level type. This problem has been resolved.

[RTI Issue ID ROUTING-149]

5.10 When Saving XML Configuration, New Filter Level not Saved if Modified Remotely

When saving *Routing Service's* XML configuration, the *Distributed Logger* configuration was saved, but if the filter level had been modified this was not reflected in the saved XML configuration. Instead, the initial parsed value was used.

[RTI Issue ID DISTLOG-19]

5.11 XML Save Functionality did not Replace Predefined Entities by Their Names

If the *Routing Service* XML configuration contained an XML predefined entity name, such as **<** or **>** inside a <content_filter> tag and a request was sent to *Routing Service* to either:

1. save the configuration to a file, or
2. view the configuration in *RTI Admin Console,*

instead of saving/sending the XML as read, *Routing Service* converted the predefined entity name to the corresponding predefined entity character, such as **<** or **>**. This led to invalid XML syntax that could not be reloaded by *Routing Service* or viewed in *Admin Console*.

For example, suppose you had the following valid XML:

```
<content_filter>
    <expression>value &gt; 100 OR value &lt; 32 </expression>
</content_filter>
```

This would have been converted to the following invalid XML before being saved to a file or sent to *Admin Console*:

```
<content_filter>
    <expression>value > 100 OR value < 32 </expression>
</content_filter>
```

This problem has been resolved.

[RTI Issue ID ROUTING-189]

5.12 Failure to Close Files with XML Snippet used to Update Routing Service

Any file containing an XML snippet that was used to update *Routing Service* was not closed after the update operation, causing a memory leak. This problem has been resolved.

[RTI Issue ID ROUTING-212]

5.13 'Get Key Value' Error and Failure to Propagate Dispose Message for Sample in Some Cases

If *Routing Service* was configured to propagate dispose messages, and the dispose message was the first message ever received from a specific instance, *Routing Service* did not propagate the dispose message. You may have seen a "get key value error." This problem has been resolved.

[RTI Issue ID ROUTING-133]

5.14 Delay in Publishing Entity Data Topic for Routes Created from Auto Routes

In some cases, *Routing Service* waited for certain events to occur before it would publish an entity data topic that described the routes created from an auto route. These events included the creation of the route's input or output, or receipt of a command that changed the route's description, such as a Pause command.

You may have noticed this behavior in *RTI Administration Console*. For example, after sending a Pause command, several new routes may have suddenly appeared.

[RTI Issue ID ROUTING-138]

5.15 Host Free Swap Memory and Total Swap Memory Reported as Zero—Windows Systems Only

When monitoring was enabled, host information, including free swap memory and total swap memory, was reported as zero on Windows platforms. This issue has been resolved. Now the swap information is sent and computed as virtual memory minus physical memory from the values provided by the Windows API GlobalMemoryStatusEx.

[RTI Issue ID ROUTING-154]

5.16 Failure to Enable Remote Administration when Loading Java 7 Adapters

Users running *Routing Service* adapters with Java 7 while remote administration was enabled may have seen the following error message:

```
ROUTERJNIGlobal_attachCurrentThread:!attach native thread to JVM
```

After reporting this error, *Routing Service* failed to enable remote administration but would continue to run. This problem has been resolved.

[RTI Issue ID ROUTING-170]

5.17 Incorrect Monitoring Data Reported when Statistics Sampling Period Equal or Longer than Publication Period

Applications subscribing to *Routing Service's* monitoring topics may have received status samples in which some fields had a value of zero. This happened periodically (some samples were correct, some were not) when **statistics_sampling_period** was equal or larger than **status_publication_period**.

This problem has been resolved. Now when **statistics_sample_period** is longer than the **status_publication_period**, *Routing Service* may publish the exact same statistics two or more times in a row—until *Routing Service* re-samples the statistics.

[RTI Issue ID ROUTING-203]

5.18 Routing Service did not notify StreamWriters of Disposed or Unregistered DDS Instances

When a DDS input receives a NOT_ALIVE_DISPOSE or NOT_ALIVE_NO_WRITERS message, it can propagate it to the route's transformation (if any) and the output. You can configure this behavior when defining a <topic_route> by using the tag <propagate_dispose> or <propagate_unregister>.

However, when you use a <route> with a <dds_input> and an adapter-based non-DDS <output>, *Routing Service* did not propagate disposed/unregistered samples, and there is no tag to enable that.

This problem has been resolved. Now *Routing Service* will always propagate dispose/unregister samples in the previous situation, so that StreamWriters can get notified when a DDS instance has been disposed or unregistered.

[RTI Issue ID ROUTING-215]

5.19 Parser did not Report Error if <domain_id> not in <administration> Tag

The tag <domain_id> is required within <administration> according to the documentation and the XSD, but failing to specify it did not generate a parsing error in the previous release. This issue did not affect the execution of *Routing Service*. If <domain_id> was not specified, the domain ID was set to the default value of 0. This problem has been resolved; if the <domain_id> tag is not found within <administration>, the parser will fail and report an error message.

[RTI Issue ID ROUTING-86]

5.20 Minor Errors in Default Configuration File, RTI_ROUTING_SERVICE.xml

The default configuration file, **RTI_ROUTING_SERVICE.xml**, included some incorrect filters.

These filters should have prevented *Routing Service* from creating topic routes for *RTI Distributed Logger* Topics. As a consequence, in some cases *Routing Service* created superfluous topic routes for *Distributed Logger* topics. This problem has been resolved; now *Routing Service* will exclude the correct topics.

[RTI Issue ID ROUTING-137]

5.21 Inconsistent QoS Settings for Remote Administration and Monitoring Entities if Loaded QoS Profile was Marked is_default_qos=true

If there was a QoS profile marked **is_default_qos="true"** in the **USER_QOS_PROFILES.xml** file, *Routing Service* would use this profile to create the DDS entities for remote administration and monitoring. This behavior resulted in inconsistent profiles because *Routing Service* would overwrite some default QoS values for the administration and monitoring entities.

In this release, *Routing Service* will ignore any profile marked as **is_default_qos="true"**.

You can still specify a profile in **USER_QOS_PROFILES.xml**. For example, the following configuration allows you to use a DataWriter QoS profile named "MyProfile" in the QoS library "MyLibrary" for the remote administration DataWriter:

```
<routing_service name="MyRoutingService">
    ...
    <administration>
        ...
        <datawriter_qos base_name="MyLibrary::MyProfile" />
    </administration>
    ...
</routing_service>
```

[RTI Issue ID ROUTING-114]

6 Known Issues

6.1 Limitations in Adapter API

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

[RTI Issue ID ROUTING-222]

6.2 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 IDROUTING-223]

6.3 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 IDROUTING-224]

6.4 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]

7 Available Documentation

Routing Service documentation includes:

- **Release Notes** (RTI_Routing_Service_ReleaseNotes.pdf)—Describes system requirements and compatibility, as well as any version-specific changes and known issues.

- **Getting Started Guide** (RTI_Routing_Service_GettingStarted.pdf)—Highlights the benefits of *Routing Service*. It provides installation and startup instructions, and walks you through several examples so you can quickly see the benefits of using *Routing Service*.
- **User's Manual** (RTI_Routing_Service_UsersManual.pdf)—Describes how to configure *Routing Service* and use it remotely.