RTI Routing Service

Release Notes

Version 5.0.0

Your systems. Working as one.

Printed in U.S.A. First printing.
August 2012.

Trademarks

Real-Time Innovations, RTI, and Connext are trademarks or registered trademarks of Real-Time Innovations, Inc. All other trademarks used in this document are the property of their respective owners.

Copy and Use Restrictions

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	pentiumInty10.0.0.0.pcx86[1]
Linux®	CentOS 5.4, 5.5 (2.6 kernel)	i86Linux2.6gcc4.1.2 x64Linux2.6gcc4.1.2
	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, 6,1	i86Linux2.6gcc4.4.5 x64Linux2.6gcc4.4.5
	Ubuntu® Server 10.04 (2.6 kernel)	i86Linux2.6gcc4.4.3 x64Linux2.6gcc4.4.3
Solaris™	Solaris 2.10	i86Sol2.10gcc3.4.4
Windows®	Windows 7 Windows Server® 2008 R2 Windows 2003 Windows Vista® Windows XP Professional	i86Win32VS2005 i86Win32VS2008 i86Win32VS2010 x64Win64VS2005 x64Win64VS2008 x64Win64VS2010

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

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

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

With the built-in *Connext* adapter, *Routing Service* can be used to forward and transform data between *Connext* applications built with *Connext*, as well as *RTI Data Distribution Service* 4.5[b-e], 4.4d, 4.3e, and 4.2e.

Note: *RTI 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]

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**

3.1.3 Mixing Different RTI Data Distribution Service Versions

This *Routing Service* version can be used simultaneously with applications built using compatible versions of *Connext* and *RTI Data Distribution Service* (see Connext Compatibility (Section 3.1)), unless special configuration is required as described above.

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*.

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

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_RoutingServiceTransformationPlugin_CreateFcn()**	This is the entry-point function.
RTITransformationClass_unloadFnc()	Function declaration: **RTI_RoutingServiceTransformationPlugin_DeleteFcn()** Member in Plugin struct: **transformation_plugin_delete**	
RTITransformationClass_createFnc()	Function declaration: **RTI_RoutingServiceTransformationPlugin_CreateTransformationFcn()** Member in Plugin struct: **transformation_plugin_create_transformation**	
RTITransformationClass_deleteFnc()	Function declaration: **RTI_RoutingServiceTransformationPlugin_DeleteTransformationFcn()** Member in Plugin struct: **transformation_plugin_delete_transformation**	

2.0.0 API (rti_routingservice.h)	2.0.1 and Higher API (routingservice_transformation.h)	Comments
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 5.0.0

4.1 New Remote Command to Get Current Configuration

There is a new remote command, **get *<target_routing_service>***, which retrieves the currently configuration and provides it in string format.

4.2 Support for Environment Variables in XML Configuration Files

This new feature allows you to refer to an environment variable within an XML tag. When the *Connext* XML parser parses the configuration file, it will expand the environment variable. To refer to an environment variable, use the format $(MY_VARIABLE).

For example:

```
<element>
    <name>The name is $(MY_NAME)</name>
    <value>The value is $(MY_VALUE)</value>
</element>
```

4.3 Ability to Propagate Related Sample Identity for Applications Using New Request-Reply Communication Pattern

RTI Connext 5.0.0 includes a new API that supports a Request-Reply communication pattern. (Only available in *Connext Messaging*. See the *Core Libraries and Utilities Release Notes* and *User's Manual* for details.)

There are new fields in the SampleInfo and WriteParams structures related to the new Request-Reply communication pattern. *Routing Service* has been updated so it can propagate these new fields.

When <publish_with_original_info> is set to TRUE in a routing service route with *Connext* output, the *Connext* output will publish samples with a **related_sample_identity** equal to that of the received SampleInfo.

Note: There is currently a scalability limitation. Repliers that communicate directly with multiple Requesters can efficiently minimize network traffic by sending a reply only to the one Requester that requested it. However, when they communicate through a routing service route, *Routing Service* receives all the replies, even for Requesters that are not on the other side of the route.

4.4 Remote Administration of Routing Service by Applications using a Requester

RTI Connext 5.0.0 includes a new API that supports a Request-Reply communication pattern (only available in *Connext Messaging*; see the *Core Libraries and Utilities Release Notes* and *User's Manual* for details). *Connext* Requesters and Repliers can communicate through *Routing Service*.

The remote administration server in *Routing Service* has been updated to act as a Replier. The pre-existing administration topics in *Routing Service* (rti/routing_service/administration/command_request and rti/routing_service/administration/command_response) can now act as requests and replies, respectively.

The new Requester object makes it extremely simple to write a user application that remotely controls *Routing Service*. See the *Routing Service User's Manual* (Section 5.3, Accessing Routing Service from a Connext Application) for example code.

4.5 Integrated Support for Distributed Logger

The *RTI Distributed Logger* library is now included with *RTI Routing Service*.

When you enable the *Distributed Logger* library, *Routing Service* will publish its log messages to *Connext*. Then you can visualize the log message data with *RTI Monitor*, a separate GUI application that can run on the same host as your application or on a different host. Since the data is provided in a *Connext* topic, you can also use *rtiddsspy* or even write your own visualization tool.

For details on how to enable the *Distributed Logger* library, see the chapter on *Configuring Routing Service* in the *RTI Routing Service User's Manual*, as well as the *RTI Distributed Logger Getting Started Guide*. These documents will show you how to use the new XML configuration tag, <distributed_logger>.

4.6 New 'paused' Field for Route Description Topics

The topics published by *Routing Service* to describe a route or an auto route (rti/routing_service/monitoring/route_data and rti/routing_service/monitoring/auto_route_data) have a new boolean field, **paused**. This field indicates if a route or auto route has been paused with the remote command pause.

4.7 Monitoring Topic Now Publishes System-Level CPU and Memory Information

In addition to publishing its own process CPU and memory usage, *Routing Service* now also publishes the CPU and memory usage for the host on which it is running. This information is in the rti/routing_service/monitoring/routing_service_status_set topic.

4.8 Support for Extensible Types

Routing Service includes partial support for the "*Extensible and Dynamic Topic Types for DDS*" specification from the Object Management Group (OMG). See Section 2.7 in the *Routing Service User's Manual* for details.

4.9 Propagation of Service Version as a DomainParticipant Property

In this release, the Routing Service version number is propagated as a DomainParticipant property called "rti.service.version". The format of the value is as follows:

```
<major>.<minor>.<release>.rev<revision>
```

The version property is set in all the DomainParticipants created by the service.

5 What's Fixed in 5.0.0

5.1 Configuration XML Schema Incorrectly Required Content Filter Parameters

There was an error in the XSD file used in XML editors for syntax validation. Because of this error, when using an XML editor, you were required to have at least one child <parameter> when specifying a <content_filter>. Parameters to a content filter are optional. This issue did not affect the execution of *Routing Service*. It only affected users that included the XSD file in their XML editors.

[RTI Issue ID ROUTING-64, Bug # 13848]

5.2 Compiler Errors for Java Applications using Remote-Administration Topics

Because the Java keyword '**final**' was used as a field name in the topics for remote administration (**RoutingServiceAdministration.idl**), Java applications that used these topics to remotely administer Routing Service failed to compile. This problem has been resolved by changing the field name to **is_final**.

[RTI Issue ID ROUTING-61, Bug # 13486]

5.3 Save Command did not Save Configurations in RTI_ROUTING_SERVICE.xml

The remote command, **save**, did not save the <routing_service> configuration tags in the file **RTI_ROUTING_SERVICE.xml** and may have generated a configuration file that was empty except for <dds> tags. This problem has been resolved; now when a tag in **RTI_ROUTING_SERVICE.xml** is being used, it will be saved when *Routing Service* receives a **save** command.

[RTI Issue ID ROUTING-82]

5.4 Monitoring Failed in Routing Service if <memory_management> Tag Used

When enabling Monitoring in *Routing Service*, *Connext* would fail to publish information about a *DataWriter* if the tag <memory_management> was included in the same participant where monitoring was enabled. This problem has been resolved.

[RTI Issue ID ROUTING-90]

5.5 Using 'disable' Command after Unloading a Configuration Resulted in Crash

Routing Service could have crashed after a 'disable' remote command if an XML configuration was not loaded at that moment (e.g., after sending the 'unload' command or after a parsing failure when issuing the 'load' command). This problem has been resolved. Now the 'disable' command will be safely ignored if there is no current configuration loaded.

[RTI Issue ID ROUTING-127]

6 Known Issues

6.1 Limitations in Adapter API

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

6.2 Sequences of Transformations in a Route are Not Supported

The tag <transformation_sequence> within a <topic_route> is not supported. In this version only one transformation per route is supported.

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*.

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.

8 Custom Supported Platform

Routing Service is also supported on the platforms listed in Table 8.1; 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 8.1 **Custom Supported Platforms**

Operating System	CPU	Compiler	RTI Architecture Abbreviation
Mistral Linux Kernel 2.6.32	ARMv7	Sourcery G++ Lite 2009q3-67 gcc 4.4.1	armv7leLinux2.6gcc4.4.1