RTI Routing Service

Release Notes

Version 5.3.0

Your systems. Working as one.

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®	All Linux platforms in the *RTI Connext DDS Core Libraries Platform Notes* for the same version number, except not supported on SUSE® 11, Wind River® Linux 4, or any custom target Linux platform.	
OS X®	All OS X platforms in the *RTI Connext DDS Core Libraries Platform Notes* for the same version number.	
QNX® [2]	Only supported on the QNX 6.5 custom target platform (ppce500v2QNX6.5.0qcc_cpp4.4.2).	
Windows®	All Windows platforms 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
QNX 6.5	PPC (e500v2)	qcc 4.4.2	ppce500v2QNX6.5.0qcc_cpp4.4.2

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 Servic*e. 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. 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.

3.5 Adapter C API Compatibility

Adapters built with the C API from *RTI Routing Service SDK* versions prior to 5.3.0 are no longer compatible due to some changes in the StreamInfo class. To reuse an existing adapter implementation, you must recompile your adapter using the new SDK headers and libraries.

4 What's New in 5.3.0

4.1 New Platforms

This release adds support for these platforms:

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

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

4.2 Support for Stream Partition and Extended DOMAIN_MATCH Creation Mode to Match PartitionQosPolicy in Addition to Topic Name in TopicRoutes

The *Routing Service* Stream concept has been extended to include partitions, with the same characteristics of the DDS PartitionQosPolicy. If your adapter supports the partition concept, it can now propagate it upstream to *Routing Service* as part of the discovery information contained in StreamInfo objects, provided by the discovery built-in StreamReaders.

Routes leverage the extended information so the DOMAIN_MATCH creation mode includes the partition settings to determine if a discovered Stream matches with the input or output of a Route. Partition matching rules apply between the partition settings of the discovered Streams and the Route's input StreamReader and output StreamWriter. Partition matching applies after a successful Stream name match. The following creation modes incorporate the partition criteria:

- ❏ ON_DOMAIN_MATCH
- ❏ ON_ROUTE_AND_DOMAIN_MATCH
- ❏ ON_ROUTE_OR_DOMAIN_MATCH

The DDS adapter has been extended to provide partition information to *Routing Service*. Consequently, A TopicRoute with a DOMAIN_MATCH based creation mode will match its input DataReader and output DataWriter when both a Topic name and partition match occur. Partition matching rules are specified by the PartitionQosPolicy.

4.3 New Java 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.

4.4 Improvements in the Java Administration API

This release includes the following improvements to the Java Administration API:

- ❏ Object lifecycle improvements:
  - RoutingServiceProperty no longer needs to be deleted and behaves as a regular Java object.
  - RoutingService implements java.lang.AutoCloseable to allow the use of try-with-resources and simplify its lifecycle management.
- ❏ Naming conventions
  - The API names have changed to use camelCase instead of underscores to follow Java conventions and to be more aligned with the OMG Java PSM and other recent RTI APIs, like the Java request-reply API.

- For the same reasons, some functions now throw Java standard exceptions instead of custom exceptions.

These changes will require some modifications in previous applications to compile.

4.5 Service will Ignore XML-Application Tags in Configuration

Routing Service will ignore any XML-Application tags that are in the loaded XML configuration. Previous versions would fail to load the XML configuration.

4.6 Support for Remote Shut Down

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 either from the supplied executable included in the RTI Connext DDS distribution or as library linked to your application.

4.7 Property to Configure Verbosity of Security0Related Log Output from TCP Transport

This is a new TCP Transport property (in the NDDS_Transport_TCPv4_Property_t structure) called **security_logging_verbosity_bitmap**. This bitmap specifies the verbosity of security-related log messages generated by the TCP Transport. For details, see the *RTI Core Libraries User's Manual*.

4.8 Enable TopicRoute Monitoring Failed If Content Filter Expression Length Longer than Maximum

TopicRoute monitoring failed to initialize if the input content filter expression was longer than RTI::RoutingService::Monitoring::EXPRESSION_MAX_LENGTH, which prevented the TopicRoute from being enabled.

This behavior has changed so monitoring will truncate the value reported in **content_filter_expression** and continue with the initialization. *Routing Service* will output a warning indicating the TopicRoute for which the monitoring value is truncated.

4.9 Service can Enable Entities even if Monitoring Fails to Initialize

Routing Service will proceed to enable an entity even if monitoring failed to initialize. This behavior can be modified with a new tag available within the general monitoring section, as follows:

```
<monitoring>
    <ignore_initialization_failure>false</ignore_initialization_failure>
<monitoring>
```

Previous versions failed to enable an entity if monitoring failed to initialize.

4.10 DDS Input's DataReader and Output's DataWriter are Created with a Default EntityNamePolicyQoS with Name based on their Configuration Names

The DDS adapter has been modified to create the input's DataReader and output's DataWriter with a default name in EntityNameQosPolicy, which consists of the parent TopicRoute's fully qualified name relative to the service.

For example, for the following configuration:

```
<routing_service name="service">
    <domain_route name="domain_route">
        <session name="session">
            <topic_route name="topic_route">
```

```
                    <input>....</input>
                    <output>...</output>
...
```

The declared input's DataReader and output 's DataWriter will be created with a name in EntityNameQosPolicy of value:

```
domain_route::session::topic_route
```

The default name is used unless you explicitly specify a different setting in the EntityNameQosPolicy configuration.

4.11 Added RTI::RoutingService_finalize_globals() to Service Library API

The Service Library API has been extended with an operation that allows finalizing global resources that Routing Service requires to operate.

For example, in the C API:

```
DDS_Boolean RTI_RoutingService_finalize_globals();
```

This operation releases resources specific to Routing Service only. RTI Connext DDS global state shall be released separately through **DomainParticipantFactory::finalize_instance()**. It should be called by your application only upon exit, after all service instances have been deleted. Calling it at a different time may cause the application to crash.

This operation is not thread safe: applications are not allowed to call this operation concurrently.

The operation returns true on success or false otherwise.

4.12 Support for Native Heap Monitoring

Routing 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*.

4.13 Support for Session as a Thread Pool

Routing Service has extended the Session model to use a thread pool to process all its contained Routes. A Session has now the capability to use up to N threads to process TopicRoutes in a concurrent, dynamic and safe manner. You can enable the thread pool by using the following tag within a session:

```
<thread_pool>
    <!-- number of threads -->
    <size>3</size>
    ...
</thread_pool>
```

The above snippet defines a thread pool of N=3 threads. See the *Routing Service User's Manual* for more information.

4.14 Support for User-Defined Sample Flags Propagation with DDS Adapter

The DDS adapter has been enhanced to propagate the SampleFlag bits of the forwarded samples. Only the user-defined sample flags are propagated.

5 What’s Fixed in 5.3.0

5.1 Dispose Instances Incorrectly Provided to Input Samples for a Transformation

Routing Service incorrectly provided the key value of disposed instances to the input samples for a transformation if the input sample sequence contained two or more disposed instances. In this situation, all the disposed instances contained the key value corresponding to the last disposed instance. This problem has been resolved.

[RTI Issue ID ROUTING-143]

5.2 Save Command Failed to Overwrite an Existing File (Windows Only)

Routing Service failed to execute a save command if the configured saved path pointed to an existing file. This problem has been resolved and *Routing Service* will overwrite the already existing file.

[RTI Issue ID ROUTING-144]

5.3 Possible Crash During Route Creation Due to Insufficient Memory

Routing Service may have crashed due to insufficient memory while creating an internal DataReader. This problem has been resolved.

[RTI Issue ID ROUTING-265]

5.4 Some TopicRoute/AutoTopicRoute Settings not Saved Correctly

The following XML tags for a TopicRoute or AutoTopicRoute were not correctly generated if a Save or Get remote command was received:

- ❏ <propagate_dispose>
- ❏ <propagate_unregister>
- ❏ <publish_with_original_info>

This problem has been resolved. Now these tags will be generated if they appeared in the original XML configuration loaded by the service.

[RTI Issue ID ROUTING-304]

5.5 Unnecessary Parse Warning

In some scenarios, *Routing Service* reported the following warning in the standard output:

```
ROUTERCfgFileAbstractRoute_setUp:Parse error at line 1463: simple route
optimization cannot be enabled.
Functionality will be automatically disabled
```

This warning was not relevant to users and has been removed.

[RTI Issue ID ROUTING-343]

5.6 Crash after Updating a Disabled AutoTopicRoute/AutoRoute with Status Monitoring Enabled

Routing Service crashed after receiving a remote command to update an AutoTopicRoute/AutoRoute that was in a disabled state and had status monitoring enabled. This problem has been resolved.

[RTI Issue ID ROUTING-389]

5.7 Memory Leak after Deleting Disabled AutoTopicRoute with Monitoring Enabled

There was a memory leak if *Routing Service* enabled monitoring and a disabled AutoTopicRoute was deleted as part of the service shutdown or by remote command. This problem has been resolved.

[RTI Issue ID ROUTING-392]

5.8 Service Reported Inaccurate Output Rate of TopicRoute

Routing Service monitoring inaccurately reported the output bytes per second of a TopicRoute, a Route with a DDS input, or a Route with a DDS output. The reported value, contained in the field **output_bytes_per_s** of DomainRouteStatusSet, SessionStatusSet, RouteStatusSet and AutoRouteStatusSet, overestimated the real value of the metric. This problem has been resolved.

[RTI Issue ID ROUTING-397]

5.9 Participant's Memory Management Settings not Applied to AutoTopicRoutes with Default QoS

If an AutoTopicRoute was configured with default input and output QoS (no reader and writer QoS was explicitly set), the memory management settings within a Participant configuration were not applied to the AutoTopicRoute. This behavior caused problems in scenarios involving large data, such as unbounded types. This problem has been resolved.

[RTI Issue ID ROUTING-399]

5.10 Routing Service did not Discover Already Present Remote Administration Clients

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

[RTI Issue ID ROUTING-402]

5.11 High CPU Usage if TopicRoute could not be Enabled after Update Remote Command

If a remote command was sent to update a TopicRoute, *Routing Service* caused high CPU usage if there was an error enabling either the input or the output after the update command. The high CPU usage was caused by an infinite loop, which could be observed by the following error messages being printed repeatedly:

```
...
ROUTERTopicRoute_enableInput:!enable stream reader
ROUTERTopicRoute_processEvent:!enable route input
ROUTERTopicRoute_onConditionTriggered:!process event
```

This problem has been resolved.

[RTI Issue ID ROUTING-406]

5.12 Possible Segmentation Fault while Updating Remote Administration Configuration

Routing Service may have issued a segmentation fault if it ran out of memory while updating the remote administration configuration. This problem has been resolved.

[RTI Issue ID ROUTING-416]

5.13 Potential High CPU Usage while TopicRoutes in Paused State

Routing Service may have entered a high CPU usage mode after a remote command was sent to pause one or more TopicRoutes and data was received by any of them. This problem has been resolved.

[RTI Issue ID ROUTING-417]

5.14 Potential Crash if DDS Adapter Logged a Write Error

A crash may have occurred if a DDS write error occurred due to an attempt to log a message and one of its formatting parameters was NULL. In this case, most of the platforms would display the following error, where *null* represents the invalid log formatting parameter that could have caused a crash on certain platforms:

```
ROUTERDdsStreamWriter_writeSample:!(null) (retcode: 3)
```

[RTI Issue ID ROUTING-420]

5.15 Crash if Adapter Discovery Listeners Notified before Service Retrieved Input/Output Discovery StreamReaders

Routing Service crashed if any of the adapter discovery listeners were notified before the service retrieved the input/output discovery StreamReaders through the **getInputStreamDiscoveryReader()** and **getOutputStreamDiscoveryReader()** APIs. This problem has been resolved so the listeners can be called from any context.

[RTI Issue ID ROUTING-437]

5.16 Crash if Transformation Plugin Failed to Load

Routing Service crashed if a Transformation plug-in failed to load. This situation occurred on route creation during startup or from a remote command.

For instance, if the configuration contained an invalid transformation library path, *Routing Service* would output the following errors before crashing:

```
ROUTERPlugin_load:!load plugin shared library:
ROUTERPlugin_load:myTransformationLib.so
ROUTERPlugin_getInstance:!load shared library
ROUTERTransformation_initialize:!get TransformationPlugin instance
ROUTERTransformation_new:!init transformation class object
ROUTERStreamPort_loadTransformation:!create transformation
ROUTERStreamReader_loadTransformation:!load transformation
ROUTERStreamReader_initialize:!load transformation
ROUTERStreamReader_new:!init ROUTERStreamReader object
ROUTERTopicRoute_initialize:!create route input
```

This problem has been resolved so the creation of a route will fail and the service will remain in a consistent state.

[RTI Issue ID ROUTING-447]

5.17 Crash if Session with AutoTopicRoute or AutoRoute was Created Remotely

Routing Service crashed if it received a remote commando to create an enabled Session containing at least one AutoTopicRoute or AutoRoute that immediately matched a stream upon creation. This problem has been resolved.

[RTI Issue ID ROUTING-451]

5.18 Wrong Version Check for Adapter and Transformation Plug-ins

Routing Service may have displayed the following warning upon loading an adapter or transformation plug-in:

```
ROUTERPlugin_checkVersion:Plugin 'MyAdapterPlugin' built with a different
major version of RTI Routing Service
ROUTERPlugin_checkVersion:Current version is 5.2.7; the plugin was built
with 1.0.0
```

This was caused by comparing the service version against the plug-in version. This warning reflects an invalid comparison given that the plug-in version is independent of the version of the service.

This problem has been resolved so *Routing Service* uses the correct version member of the loaded plugin that indicates the service version it was built against. You still may see the warning and in that case it will be accurate.

[RTI Issue ID ROUTING-469]

5.19 Failure to Parse Types where a Valuetype Inherited from a Struct

Routing Service did not parse types in which a valuetype inherited from a struct.

For *Connext DDS*, an IDL valuetype and an IDL struct are the same and interchangeable, so the inheritance relationship should have been allowed. *Code Generator* successfully generated code for such an IDL valuetype.

This problem has been resolved.

[RTI Issue ID ROUTING-477]

6 Known Issues

6.1 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]

6.2 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]

6.3 Limitations in Adapter API

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

[RTI Issue ID ROUTING-222]

6.4 Tranformation Plugin from rtirsassigntransf does not Support Unions

The library distributed with *Routing Service*, rtirsassigntransf, does not support types that contain Unions. If your type is a Union or has Union members, you may run into errors such as:

```
...
DDS_DynamicData_delete:invalid operation, bound to member id 4
ROUTERTransformation_transform:transformation
'::Transformations::Assign' error (0): Unbinding error
ROUTERStreamWriter_write:!transform samples
...
```

[RTI Issue ID ROUTING-429]

6.5 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]