RTI Routing Service

Version 7.7.0

Chapter 1

RTI Routing Service

RTI Routing Service∗ supports the integration and scaling of disparate and geographically dispersed systems. Using off-the-shelf or custom developed plugins, *Routing Service* supports the definition of data transformations and adapters that can interface to multiple protocols in addition to DDS, such as MQTT or legacy code written to the network socket API.

You can learn how to use *Routing Service* through the `RTI Routing Service User's Manual`.

A reference to the API can be found here:

- **RTI Routing Service Adapter API** (p. 12)
- **RTI Routing Library API** (p. 11)

1.1 Feedback and Support for this Release

For more information, visit our knowledge base (`https://community.rti.com/kb`) to see sample code, general information on RTI Connext, performance information, troubleshooting tips, and technical details.

By its very nature, the knowledge base is continuously evolving and improving. We hope that you will find it helpful. If there are questions that you would like to see addressed or comments you would like to share, please send email to `support@rti.com`. We can only guarantee a response for customers with a current maintenance contract or subscription. To purchase a maintenance contract or subscription, contact your local RTI representative (see `http://www.rti.com/company/contact.html`), send an email request to `sales@rti.com`, or call +1 (408) 990-7400.

Please do not hesitate to contact RTI with questions or comments about this release. We welcome any input on how to improve RTI Routing Service∗ and *RTI Connext DDS* to suit your needs.

Chapter 2

Module Index

2.1 Modules

Here is a list of all modules:

Chapter 3

Hierarchical Index

3.1 Class Hierarchy

This inheritance list is sorted roughly, but not completely, alphabetically:

Chapter 4

Class Index

4.1 Class List

Here are the classes, structs, unions and interfaces with brief descriptions:

Chapter 5

File Index

5.1 File List

Here is a list of all documented files with brief descriptions:

Chapter 6

Module Documentation

6.1 RTI Routing Library API

Routing Service can be deployed as a Java library This API allows you to create, configure and start RTI Routing Service instances from your application.

Classes

- class **RoutingService**

  RTI Routing Service.
- class **RoutingServiceProperty**

  Configuration of RTI Routing Service.

6.1.1 Detailed Description

Routing Service can be deployed as a Java library This API allows you to create, configure and start RTI Routing Service instances from your application.

The following code shows the typical use of the API:

```
RoutingServiceProperty prop = new RoutingServiceProperty();
prop.cfgFile = "my_routing_service_cfg.xml";
prop.serviceName = "my_routing_service";
try(RoutingService rs = new RoutingService(prop)) {
   rs.start();
   ...
}
```

Instead of a file, you can use XML strings to configure RTI Routing Service. See **RoutingServiceProperty** (p. 37) for more information.

Your application must be built against the RTI Routing Service jar file in <**NDDSHOME**>/**lib**/**java**/ which must be able to dynamically load the RTI Routing Service library in <**NDDSHOME**>/**lib**/<**architecture**>/

Exceptions: Calls on **RoutingService** (p. 33) and **RoutingServiceProperty** (p. 37) may throw **infrastructure.RoutingServiceException** (p. 37) upon failure.

6.2 RTI Routing Service Adapter API

This module describes the Java Adapter API.

Classes

- interface **Adapter**

  Adapter *(*p. *19) interface (required).*
- interface **Connection**

  Connection *(*p. *24) interface (required).*
- interface **DiscoveryConnection**

  DiscoveryConnection *(*p. *29) interface (optional).*
- interface **Entity**

  Entity *(*p. *31) interface (required).*
- interface **Session**

  Session *(*p. *43) interface (optional).*
- interface **StreamReader**

  StreamReader *(*p. *44) interface (optional).*
- interface **StreamReaderListener**

  StreamReaderListener *(*p. *46) used to notify Routing Service that new data is available.*
- interface **StreamWriter**

  StreamWriter *(*p. *47) interface (optional).*

6.2.1 Detailed Description

This module describes the Java Adapter API.

Adapters are pluggable components that allow *RTI Routing Service* to consume and produce data for different data domains (e.g., Connext DDS, MQTT, Socket, etc.).

By default, *Routing Service* is distributed with a builtin DDS adapter. Any other adapter plugins must be provided as external components registered through the XML configuration or through the Library API.

The following figure shows an overview of the *Routing Service* adapter.

Input adapters are used to collect data samples from different data domains, such as DDS or MQTT. The input samples are processed by the Routing Service engine and are passed along to output adapters through the Processor, applying any Transformation beforehand if present.

For additional details about Adapter configuration see the `RTI Routing Service User's Manual`.

6.2.2 Development Requirements

	Linux/macOS Systems	Windows Systems
JAR	rtirsadapter.jar	
	dds.jar	
Native Libraries	librtirsjniadapter.so	rtirsjniadapter.dll
	librtirsinfrastructure.so	rtirsinfrastructure.dll
	libndds.so	nddsc.dll
	libnddscore.so	nddscore.dll

6.2.3 Architecture

The Adapter architecture is shown in the class diagram below.

The sequence diagram in this figure shows when the different adapter entities are created.

- An **com.rti.routingservice.adapter.Adapter** (p. 19) object is created as soon as RTI Routing Service starts and finds out about the plug-ins that will be used by underlying DomainRoute connections.
- A **com.rti.routingservice.adapter.Connection** (p. 24) object is created when the DomainRoute (<domain_↩ route>) that contain it is enabled.
- A **com.rti.routingservice.adapter.Session** (p. 43) object is created when the associated service Session (<session>) is enabled.
- An inputs **com.rti.routingservice.adapter.StreamReader** (p. 44) is created if the Route is enabled and the creation mode condition associated with the <input> tag becomes true.
- An outputs **com.rti.routingservice.adapter.StreamWriter** (p. 47) is created if the route is enabled and the creation mode condition associated with the <output> tag becomes true.

6.2.3.1 Stream Discovery

A Route cannot forward data until the type representations (e.g., TypeCode) associated with the input and output streams are available.

If a Route refers to types that are not defined in the configuration file, RTI Routing Service has to discover their type representation (e.g., TypeCode) before creating the **com.rti.routingservice.adapter.StreamReader** (p. 44) and **com.↩ rti.routingservice.adapter.StreamWriter** (p. 47). The adapter discovery API is used to provide stream and type information in a data domain to Routing Service∗.

The discovery API consists of the following methods:

- **com.rti.routingservice.adapter.DiscoveryConnection::getInputStreamDiscoveryReader** (p. 30)
- **com.rti.routingservice.adapter.DiscoveryConnection::getOutputStreamDiscoveryReader** (p. 30)

These methods provide access to **com.rti.routingservice.adapter.StreamReader** (p. 44) used to discover streams in the data domain associated with a connection.

The input stream discovery **com.rti.routingservice.adapter.StreamReader** (p. 44) provides information about input streams. An input stream is a stream from which an input's **com.rti.routingservice.adapter.StreamReader** (p. 44) reads data. Notification of disposed scenarios, where an input stream disappears, are also made using the input stream discovery StreamReader.

In the builtin DDS adapter, the input stream discovery StreamReader is associated with the publication builtin Data↩ Reader of the DomainParticipant.

The output stream discovery **com.rti.routingservice.adapter.StreamReader** (p. 44) provides information about output streams. An output stream is a stream to which an output's **com.rti.routingservice.adapter.StreamWriter** (p. 47) can write data. Notification of disposed scenarios, where an output stream disappears, are also made using the output stream discovery StreamReader.

In the builtin DDS adapter, the output stream discovery StreamReader is associated with the subscription builtin Data↩ Reader of the DomainParticipant.

The samples provided by the stream discovery StreamReaders have the type com.rti.routingservice.infrastructure.↩ StreamInfo.

6.3 RTI Routing Service

RTI Routing Service.

Modules

- **RTI Routing Library API**

  Routing Service can be deployed as a Java library This API allows you to create, configure and start RTI Routing Service instances from your application.
- **RTI Routing Service Adapter API**

  This module describes the Java Adapter API.
- **RTI Routing Service Infrastructure**

  This module contains common definitions used across other functional modules.

6.3.1 Detailed Description

RTI Routing Service.

6.4 RTI Routing Service Infrastructure

This module contains common definitions used across other functional modules.

Modules

- **Standard Type Representation Kinds**

  Standard type Representation kinds.
- **Standard Data Representation Kinds**

  Standard data representation kinds.

Classes

- class **AdapterException**

  Adapter *(*p. *19) exception.*
- class **StreamInfo**

  Stream information.
- class **TypeInfo**

  Type information.
- class **Version**

  Represents the version number of an adapter plugin.
- class **OutOfResourcesException**

  No Memory exception.
- class **RoutingServiceException**

  Routing Library API exception.

6.4.1 Detailed Description

This module contains common definitions used across other functional modules.

6.5 Standard Type Representation Kinds

Standard type Representation kinds.

Variables

- static int **DYNAMIC_TYPE_REPRESENTATION**

  Dynamic type representation.
- static int **XML_TYPE_REPRESENTATION**

  [Not supported] *XML type representation.*
- static int **JAVA_OBJECT_TYPE_REPRESENTATION**

  [Not supported] *Java object type representation.*

6.5.1 Detailed Description

Standard type Representation kinds.

6.5.2 Variable Documentation

6.5.2.1 DYNAMIC_TYPE_REPRESENTATION

```
int DYNAMIC_TYPE_REPRESENTATION  [static]
```

Dynamic type representation.

Types with this representation are provided as RTI Connext TypeCodes.

For more information about TypeCodes, see the "Data Types and DDS Data Samples" chapter in the RTI Connext DDS Core Libraries User's Manual.

6.5.2.2 XML_TYPE_REPRESENTATION

```
int XML_TYPE_REPRESENTATION  [static]
```

[Not supported] XML type representation.

Types with this representation are provided as XML strings. The XML schema is the RTI Connext XML schema for type representation.

For more information about XML representation, see the "Data Types and DDS Data Samples" chapter in the RTI Connext DDS Core Libraries User's Manual.

6.5.2.3 JAVA_OBJECT_TYPE_REPRESENTATION

```
int JAVA_OBJECT_TYPE_REPRESENTATION  [static]
```

[Not supported] Java object type representation.

Types with this representation are provided as Java objects.

6.6 Standard Data Representation Kinds

Standard data representation kinds.

Variables

- static int **DYNAMIC_DATA_REPRESENTATION**

  Dynamic data representation.
- static int **XML_DATA_REPRESENTATION**

  [Not supported] *XML data representation.*
- static int **JAVA_OBJECT_DATA_REPRESENTATION**

  [Not supported] *Java object data representation.*

6.6.1 Detailed Description

Standard data representation kinds.

6.6.2 Variable Documentation

6.6.2.1 DYNAMIC_DATA_REPRESENTATION

```
int DYNAMIC_DATA_REPRESENTATION  [static]
```

Dynamic data representation.

Samples with this representation are provided as RTI Connext DynamicData objects.

For more information about TypeCodes, see the "Data Types and DDS Data Samples" chapter in the RTI Connext DDS Core Libraries User's Manual.

6.6.2.2 XML_DATA_REPRESENTATION

```
int XML_DATA_REPRESENTATION  [static]
```

[Not supported] XML data representation.

Samples with this representation are provided as XML strings. These samples are created according to the RTI Connext XML schema for type representation.

For more information about XML representation, see the "Data Types and DDS Data Samples" chapter in the RTI Connext DDS Core Libraries User's Manual.

6.6.2.3 JAVA_OBJECT_DATA_REPRESENTATION

```
int JAVA_OBJECT_DATA_REPRESENTATION  [static]
```

[Not supported] Java object data representation.

Samples with this representation are provided as Java objects.

Chapter 7

Class Documentation

7.1 Adapter Interface Reference

Adapter (p. 19) interface (required).

Public Member Functions

- abstract **Connection** **createConnection** (String routingServiceName, String routingServiceGroupName, **StreamReaderListener** inputStreamDiscoveryListener, **StreamReaderListener** outputStreamDiscovery↩ Listener, Properties properties) throws AdapterException

  *Creates a **com.rti.routingservice.adapter.Connection** (p. 24).*
- abstract void **deleteConnection** (**Connection** connection) throws AdapterException

  *Deletes a **com.rti.routingservice.adapter.Connection** (p. 24).*
- abstract **Version** **getVersion** ()

  Returns the version of this adapter plugin.

7.1.1 Detailed Description

Adapter (p. 19) interface (required).

Java adapters are registered with RTI Routing Service using the XML tag <java_adapter_plugin>

For example:

```
<dds>
    ...
    <plugin_library name="MyAdapterLib">
        <java_adapter_plugin name="MyAdapterPlugin">
            <class_name>com.rti.adapters.MyAdapter</class_name>
        </java_adapter_plugin>
        ...
    </plugin_library>
    ...
    <routing_service>
    ...
    </routing_service>
    ...
</dds>
```

In the above example, the interface is implemented by the com.rti.adapters.MyAdapter class.

When RTI Routing Service creates a Java adapter, it uses a constructor with one java.lang.Properties parameter. Those properties contain the configuration of the adapter and, in addition, the following special properties:

- rti.routing_service.app_name: whose value contains the given application name
- rti.routing_service.group_name: whose value represents the given group_name
- rti.routing_service.version: whose value represents the RTI Routing Service version
- rti.routing_service.verbosity: whose value represents the verbosity in use by RTI Routing Service and is one of the following strings: NONE, EXCEPTION, WARN, INFO, DEBUG.

If that constructor is not found the constructor with no parameters is used instead.

7.1.2 Member Function Documentation

7.1.2.1 createConnection()

```
abstract  Connection createConnection (
             String routingServiceName,
             String routingServiceGroupName,
              StreamReaderListener inputStreamDiscoveryListener,
              StreamReaderListener outputStreamDiscoveryListener,
             Properties properties ) throws  AdapterException  [abstract]
```

Creates a **com.rti.routingservice.adapter.Connection** (p. 24).

A **Connection** (p. 24) provides access to a data domain (such as a DDS domain or a JMS network provider).

Connection (p. 24) objects are created when the DomainRoutes that contain them are enabled.

Required: Yes

Parameters

routingServiceName	<<*in*>> The routing service execution name.
routingServiceGroupName	<<*in*>> The group name of the routing service execution.
inputStreamDiscoveryListener	<<*in*>> The listener of the built-in **StreamReader** (p. 44) that notifies the discovery of new input streams.
outputStreamDiscoveryListener	<<*in*>> The listener of the built-in **StreamReader** (p. 44) that notifies the discovery of new output streams.

Parameters

properties	<<*in*>> Configuration properties for the **Connection** (p. 24).

Returns

New **Connection** (p. 24) if successful.

Exceptions

AdapterException	- if an error occurs.

See also

com.rti.routingservice.adapter.Adapter.deleteConnection (p. 21)

7.1.2.2 deleteConnection()

```
abstract void deleteConnection (
              Connection connection ) throws  AdapterException  [abstract]
```

Deletes a **com.rti.routingservice.adapter.Connection** (p. 24).

Connection (p. 24) objects are deleted when the DomainRoutes that contain them are disabled or RTI Routing Service is closed.

Required: Yes

Parameters

connection	<<*in*>> **Connection** (p. 24) to be deleted

Exceptions

AdapterException	- if an error occurs.

See also

com.rti.routingservice.adapter.Adapter.createConnection (p. 20)

7.1.2.3 getVersion()

```
abstract  Version getVersion ( )  [abstract]
```

Returns the version of this adapter plugin.

7.2 AdapterException Class Reference

Adapter (p. 19) exception.

Inherits Exception.

Public Member Functions

- **AdapterException** (int nativeCode, String message)
  *Creates an **AdapterException** (p. 22) with a native error code and a message.*
- **AdapterException** (int nativeCode, String message, Throwable cause)
  *Creates an **AdapterException** (p. 22) with a native error, message and cause.*
- **AdapterException** (int nativeCode, Throwable cause)
  *Creates an **AdapterException** (p. 22) with a native error, and cause.*
- int **getNativeCode** ()
  Gets the native error code.

7.2.1 Detailed Description

Adapter (p. 19) exception.

Signals that an error has occurred in some of the adapter APIs.

7.2.2 Constructor & Destructor Documentation

7.2.2.1 AdapterException() [1/3]

```
AdapterException (
            int nativeCode,
            String message )
```

Creates an **AdapterException** (p. 22) with a native error code and a message.

Parameters

nativeCode	<<*in*>> The native error code.
message	<<*in*>> The error message.

7.2.2.2 AdapterException() [2/3]

```
AdapterException (
            int nativeCode,
            String message,
            Throwable cause )
```

Creates an **AdapterException** (p. 22) with a native error, message and cause.

Parameters

nativeCode	<<*in*>> The native error code.
message	<<*in*>> The error message.
cause	<<*in*>> The cause.

7.2.2.3 AdapterException() [3/3]

```
AdapterException (
            int nativeCode,
            Throwable cause )
```

Creates an **AdapterException** (p. 22) with a native error, and cause.

Parameters

nativeCode	<<*in*>> The native error code.
cause	<<*in*>> The cause.

7.2.3 Member Function Documentation

7.2.3.1 getNativeCode()

```
int getNativeCode ( )
```

Gets the native error code.

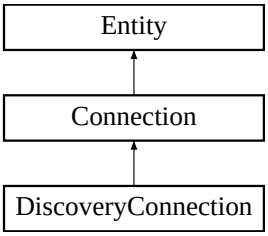
Returns

Native error code.

7.3 Connection Interface Reference

Connection (p. 24) interface (required).

Inheritance diagram for Connection:



Public Member Functions

- **Session createSession** (Properties properties) throws AdapterException

  *Creates a **Session** (p. 43).*
- void **deleteSession** (**Session** session) throws AdapterException

  *Deletes a **Session** (p. 43).*
- **StreamReader createStreamReader** (**Session** session, **StreamInfo** streamInfo, Properties properties, **StreamReaderListener** listener) throws AdapterException

  *Creates a **StreamReader** (p. 44).*
- void **deleteStreamReader** (**StreamReader** streamReader) throws AdapterException

  *Deletes a **StreamReader** (p. 44).*
- **StreamWriter createStreamWriter** (**Session** session, **StreamInfo** streamInfo, Properties properties) throws AdapterException

  *Creates a **StreamWriter** (p. 47).*
- void **deleteStreamWriter** (**StreamWriter** streamWriter) throws AdapterException

  *Deletes a **StreamWriter** (p. 47).*

7.3.1 Detailed Description

Connection (p. 24) interface (required).

A **Connection** (p. 24) object provides access to a data domain (such as a DDS domain or a JMS network provider).

In the XML configuration file, Connections are created using the tag <connection> within a DomainRoute.

7.3.2 Member Function Documentation

7.3.2.1 createSession()

```
Session createSession (
            Properties properties ) throws  AdapterException
```

Creates a **Session** (p. 43).

A **Session** (p. 43) is a concurrency unit within a **Connection** (p. 24) that has an associated set of StreamReaders and StreamWriters. Access to the StreamReaders and StreamWriters in the same **Session** (p. 43) is serialized by RTI Routing Service.

Session (p. 43) objects are created when the associated routing service sessions are enabled.

In the XML configuration file, Sessions are associated with the tag <session> within a domain route.

Required: No

The Java implementation of this method can return null if sessions are not required by the adapter.

Parameters

properties	<<*in*>> Configuration properties for the **Session** (p. 43).

Returns

New **Session** (p. 43) if successful.

Exceptions

AdapterException	- if an error occurs.

See also

com.rti.routingservice.adapter.Connection.deleteSession (p. 25)

7.3.2.2 deleteSession()

```
void deleteSession (
              Session session ) throws  AdapterException
```

Deletes a **Session** (p. 43).

Session (p. 43) objects are deleted when the routing service sessions that contain them are disabled.

Required: No

Parameters

session	<<*in*>> **Session** (p. 43) to be deleted.

Exceptions

AdapterException	- if an error occurs.

See also

com.rti.routingservice.adapter.Connection.createSession (p. 25)

7.3.2.3 createStreamReader()

```
StreamReader createStreamReader (
             Session session,
             StreamInfo streamInfo,
            Properties properties,
             StreamReaderListener listener ) throws  AdapterException
```

Creates a **StreamReader** (p. 44).

A **StreamReader** (p. 44) provides a way to read samples of a specific type from a data domain.

In the XML configuration file, StreamReaders are associated with the tag <input> within <route> or <auto_route>.

This method is called when the route is enabled and the 'creation mode' condition associated with the route's input becomes true.

Required: Only when the adapter is used to read data.

The Java implementation of this method can return null if the adapter is not used to read data.

Parameters

session	<<*in*>> **Session** (p. 43) associated with the **StreamReader** (p. 44). This parameter is null if Sessions are not used by the adapter.
streamInfo	<<*in*>> Name of the stream and type representation.
properties	<<*in*>> Configuration properties for the **StreamReader** (p. 44).
listener	<<*in*>> The listener of the **StreamReader** (p. 44) used to notify the routing service when new data is available.

Returns

New **StreamReader** (p. 44) if successful.

Exceptions

AdapterException	- if an error occurs.

See also

com.rti.routingservice.adapter.Connection.deleteStreamReader (p. 27)

7.3.2.4 deleteStreamReader()

```
void deleteStreamReader (
            StreamReader streamReader ) throws  AdapterException
```

Deletes a **StreamReader** (p. 44).

A **StreamReader** (p. 44) object is deleted when the route that contains it is disabled, when the 'creation mode' condition associated with the route's input becomes false or when RTI Routing Service is closed.

Required: Only when the adapter is used to read data.

Parameters

streamReader	<<*in*>> **StreamReader** (p. 44) to be deleted.

Exceptions

AdapterException	- if an error occurs.

See also

com.rti.routingservice.adapter.Connection.createStreamReader (p. 26)

7.3.2.5 createStreamWriter()

```
StreamWriter createStreamWriter (
            Session session,
```

```
 StreamInfo streamInfo,
Properties properties ) throws  AdapterException
```

Creates a **StreamWriter** (p. 47).

A **StreamWriter** (p. 47) provides a way to write samples of a specific type in a data domain.

In the XML configuration file, StreamWriters are associated with the tag <output> within <route> or <auto_route>.

This method is called when the route is enabled and the 'creation mode' condition associated with the route's output becomes true.

Required: Only when the adapter is used to write data. The Java implementation of this method can return null if the adapter is not used to write data.

Parameters

session	<<*in*>> **Session** (p. 43) associated with the **StreamWriter** (p. 47). This parameter is null if Sessions are not used by the adapter.
streamInfo	<<*in*>> Name of the stream and type representation.
properties	<<*in*>> Configuration properties for the **StreamWriter** (p. 47).

Returns

New **StreamWriter** (p. 47) if successful.

Exceptions

AdapterException	- if an error occurs.

See also

com.rti.routingservice.adapter.Connection.deleteStreamWriter (p. 28)

7.3.2.6 deleteStreamWriter()

```
void deleteStreamWriter (
             StreamWriter streamWriter ) throws  AdapterException
```

Deletes a **StreamWriter** (p. 47).

A **StreamWriter** (p. 47) object is deleted when the route or domain route that contains it is disabled, when the 'creation mode' condition associated with the route's output becomes false or when RTI Routing Service is closed.

Required: Only when the adapter is used to write data.

Parameters

streamWriter	<<*in*>> **StreamWriter** (p. 47) to be deleted.

Exceptions

AdapterException	- if an error occurs.

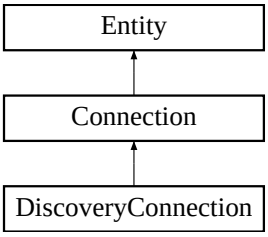
See also

com.rti.routingservice.adapter.Connection.createStreamWriter (p. 27)

7.4 DiscoveryConnection Interface Reference

DiscoveryConnection (p. 29) interface (optional).

Inheritance diagram for DiscoveryConnection:



Public Member Functions

- **StreamReader getInputStreamDiscoveryReader** () throws AdapterException
  *Gets the input stream discovery **StreamReader** (p. 44).*
- **StreamReader getOutputStreamDiscoveryReader** () throws AdapterException
  *Gets the output stream discovery **StreamReader** (p. 44).*
- Object **copyTypeRepresentation** (Object typeRepresentation) throws AdapterException
  Copies a type representation.
- void **deleteTypeRepresentation** (Object typeRepresentation) throws AdapterException
  Deletes a type representation.

7.4.1 Detailed Description

DiscoveryConnection (p. 29) interface (optional).

The **DiscoveryConnection** (p. 29) interface provides optional discovery methods for a connection.

7.4.2 Member Function Documentation

7.4.2.1 getInputStreamDiscoveryReader()

```
StreamReader getInputStreamDiscoveryReader ( ) throws  AdapterException
```

Gets the input stream discovery **StreamReader** (p. 44).

Returns a **StreamReader** (p. 44) that is used by RTI Routing Service to discover input streams. An input stream is a stream from which a **StreamReader** (p. 44) can read data. Disposed scenarios, where an input stream disappears, are also notified using this discovery
StreamReader (p. 44).

The **StreamReaderListener** (p. 46) associated with this **StreamReader** (p. 44) is provided as a parameter to the method **Adapter.createConnection** (p. 20)

7.4.2.2 getOutputStreamDiscoveryReader()

```
StreamReader getOutputStreamDiscoveryReader ( ) throws  AdapterException
```

Gets the output stream discovery **StreamReader** (p. 44).

This method returns a **StreamReader** (p. 44) that is used by RTI Routing Service to discover output streams. An output stream is a stream to which StreamWriters can write data. Disposed scenarios, where an output stream disappears, are also notified using this discovery **StreamReader** (p. 44).

The **StreamReaderListener** (p. 46) associated with this **StreamReader** (p. 44) is provided as a parameter to the method **Adapter.createConnection** (p. 20).

7.4.2.3 copyTypeRepresentation()

```
Object copyTypeRepresentation (
            Object typeRepresentation ) throws  AdapterException
```

Copies a type representation.

This method is part of the adapter discovery API and is used by RTI Routing Service to copy the type representation associated with the discovered streams.

Parameters

typeRepresentation	<<*in*>> Type representation to be copied.

Exceptions

AdapterException	- if an error occurs.

See also

com.rti.routingservice.adapter.Connection.deleteTypeRepresentation

7.4.2.4 deleteTypeRepresentation()

```
void deleteTypeRepresentation (
            Object typeRepresentation ) throws  AdapterException
```

Deletes a type representation.

This method is part of the adapter discovery API.

Parameters

typeRepresentation	<<*in*>> Type representation to be deleted.

Exceptions

AdapterException	- if an error occurs.

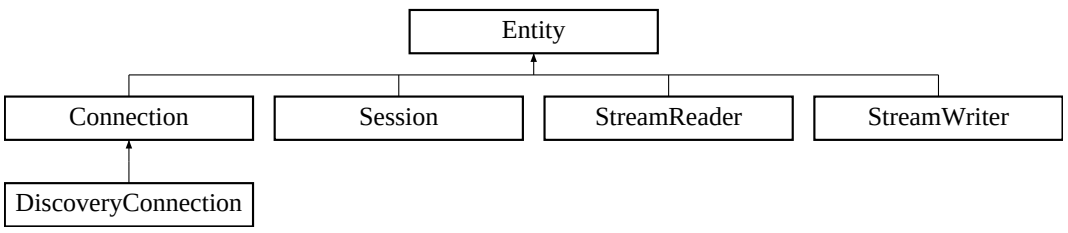
See also

com.rti.routingservice.adapter.Connection.copyTypeRepresentation

7.5 Entity Interface Reference

Entity (p. 31) interface (required).

Inheritance diagram for Entity:



Public Member Functions

- void **update** (Properties properties) throws AdapterException

  Updates the configuration of an adapter entity.

7.5.1 Detailed Description

Entity (p. 31) interface (required).

The adapter entities are:

- **Connection** (p. 24)
- **Session** (p. 43)
- **StreamReader** (p. 44)
- **StreamWriter** (p. 47)

7.5.2 Member Function Documentation

7.5.2.1 update()

```
void update (
            Properties properties ) throws  AdapterException
```

Updates the configuration of an adapter entity.

This method is called when remote administration is used.

Adapter (p. 19) entities that can be updated are:

- **Connection** (p. 24)
- **Session** (p. 43)
- **StreamReader** (p. 44)
- **StreamWriter** (p. 47)

Required: No. Implement this function only when remote configuration is needed.

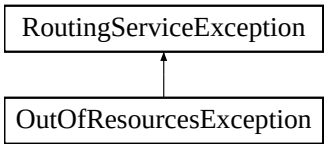
Parameters

properties	<<*in*>> New configuration properties.

7.6 OutOfResourcesException Class Reference

No Memory exception.

Inheritance diagram for OutOfResourcesException:



7.6.1 Detailed Description

No Memory exception.

Signals that a native allocation has failed

7.7 RoutingService Class Reference

RTI Routing Service.

Inherits AutoCloseable.

Public Member Functions

- **RoutingService** (**RoutingServiceProperty** property)
  Create a new RTI Routing Service instance.
- void **close** ()
  Releases native memory resources.
- void **start** ()
  Start RTI Routing Service.
- void **stop** ()
  Stop RTI Routing Service.
- void **createEntity** (String parentName, String xmlSnippet)
  Create new domain route, session, topic route or auto topic route.
- void **deleteEntity** (String entityName)
  Delete domain route, session, topic route or auto topic route.
- String **getConfiguration** ()
  Get current configuration.

Static Public Attributes

- static final int **LOG_VERBOSITY_SILENT** = 0

  Verbosity level: silent.
- static final int **LOG_VERBOSITY_EXCEPTIONS** = 3

  Verbosity level: RTI_LOG_BIT_FATAL_ERROR (0x00000001) | *RTI_LOG_BIT_EXCEPTION (0x00000002)*
- static final int **LOG_VERBOSITY_WARNINGS** = 7

  Verbosity level: RTI_LOG_BIT_FATAL_ERROR (0x00000001) | *RTI_LOG_BIT_EXCEPTION (0x00000002)* | *RTI_LOG↩_BIT_WARN (0x00000004)*
- static final int **LOG_VERBOSITY_INFO** = 31

  Verbosity level: RTI_LOG_BIT_FATAL_ERROR (0x00000001) | *RTI_LOG_BIT_EXCEPTION (0x00000002)* | *RTI_LOG↩_BIT_WARN (0x00000004)* | *RTI_LOG_BIT_LOCAL (0x00000008)* | *RTI_LOG_BIT_REMOTE (0x00000010)*

7.7.1 Detailed Description

RTI Routing Service.

7.7.2 Constructor & Destructor Documentation

7.7.2.1 RoutingService()

```
RoutingService (
            RoutingServiceProperty property )
```

Create a new RTI Routing Service instance.

Parameters

property	The properties to configure RTI Routing Service.

7.7.3 Member Function Documentation

7.7.3.1 close()

```
void close ( )
```

Releases native memory resources.

This frees native memory resources associated with the service

7.7.3.2 start()

```
void start ( )
```

Start RTI Routing Service.

This is a non-blocking operation. RTI Routing Service will create its own set of threads to perform its tasks.

7.7.3.3 stop()

```
void stop ( )
```

Stop RTI Routing Service.

This functions won't return the execution control until the instance is fully stopped.

7.7.3.4 createEntity()

```
void createEntity (
             String parentName,
             String xmlSnippet )
```

Create new domain route, session, topic route or auto topic route.

Parameters

parentName	Fully-qualified name of parent entity under which to create new entity.
xmlSnippet	XML configuration string for new entity. Read from file if string starts with file://

See also

For more information in Resource Identifiers and Fully-Qualified Names, see `Resource Indentifiers` in the User's Manual.

7.7.3.5 deleteEntity()

```
void deleteEntity (
             String entityName )
```

Delete domain route, session, topic route or auto topic route.

Parameters

entityName	Fully-qualified name of entity to delete.

See also

For more information in Resource Identifiers and Fully-Qualified Names, see `Resource Indentifiers` in the User's Manual.

7.7.3.6 getConfiguration()

```
String getConfiguration ( )
```

Get current configuration.

Returns

A string containing the service's configuration in XML

7.7.4 Member Data Documentation

7.7.4.1 LOG_VERBOSITY_SILENT

```
final int LOG_VERBOSITY_SILENT = 0  [static]
```

Verbosity level: silent.

7.7.4.2 LOG_VERBOSITY_EXCEPTIONS

```
final int LOG_VERBOSITY_EXCEPTIONS = 3  [static]
```

Verbosity level: RTI_LOG_BIT_FATAL_ERROR (0x00000001) | RTI_LOG_BIT_EXCEPTION (0x00000002)

7.7.4.3 LOG_VERBOSITY_WARNINGS

```
final int LOG_VERBOSITY_WARNINGS = 7  [static]
```

Verbosity level: RTI_LOG_BIT_FATAL_ERROR (0x00000001) | RTI_LOG_BIT_EXCEPTION (0x00000002) | RTI_↩ LOG_BIT_WARN (0x00000004)

7.7.4.4 LOG_VERBOSITY_INFO

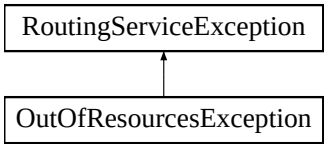
```
final int LOG_VERBOSITY_INFO = 31  [static]
```

Verbosity level: RTI_LOG_BIT_FATAL_ERROR (0x00000001) | RTI_LOG_BIT_EXCEPTION (0x00000002) | RTI_↩ LOG_BIT_WARN (0x00000004) | RTI_LOG_BIT_LOCAL (0x00000008) | RTI_LOG_BIT_REMOTE (0x00000010)

7.8 RoutingServiceException Class Reference

Routing Library API exception.

Inheritance diagram for RoutingServiceException:



7.8.1 Detailed Description

Routing Library API exception.

Signals that a problem has occurred in some of the Routing Library APIs

7.9 RoutingServiceProperty Class Reference

Configuration of RTI Routing Service.

Public Member Functions

- **RoutingServiceProperty** ()

  Constructor.

Public Attributes

- String **cfgFile**

  Path to an RTI Routing Service configuration file.
- String[ ] **cfgStrings**

  XML configuration represented as strings.
- String **serviceName**

  The name of the Routing Service instance to run.
- String **applicationName**

  Assigns a name to the execution of the RTI Routing Service.
- boolean **enforceXsdValidation** = true

  Controls whether the service applies XSD validation to the loaded configuration.
- int **serviceVerbosity** = **RoutingService.LOG_VERBOSITY_EXCEPTIONS**

  The verbosity of the service.
- int **ddsVerbosity** = **RoutingService.LOG_VERBOSITY_EXCEPTIONS**

  The verbosity of the service.
- int **domainIdBase** = 0

  Value that is added to the domain IDs of the domain routes in the XML configuration.
- String **pluginSearchPath**

  This path is used to look for plug-in libraries.
- boolean **dontStartService** = false

  *Set this to true to if you do not want RTI Routing Service enabled when **RoutingService.start** (p. 34) is called.*
- boolean **enableAdministration** = false

  Set this to true to enable remote administration or false to disable it.
- int **administrationDomainId** = 0

  *If **enableAdministration** (p. 41) is true, this is the domain ID to use for remote administration.*
- boolean **enableMonitoring** = false

  Set it to true to enable remote monitoring or false to disable it.
- int **monitoringDomainId** = 0

  *If **enableMonitoring** (p. 41) is true, this is the domain ID to use for remote monitoring.*
- boolean **skipDefaultFiles** = false

  Set it to true to avoid loading the standard files usually loaded by RTI Routing Service.
- boolean **identifyExecution** = false

  Set this to true to append the host name and process ID to the RTI Routing Service execution name.
- String **licenseFileName**

  Path to an RTI Routing Service license file.
- Properties **user_environment** = new Properties()

  Dictionary of user variables. The dictionary provides a parallel way to expand XML configuration variables in the form , when they are not defined in the environment.

7.9.1 Detailed Description

Configuration of RTI Routing Service.

7.9.2 Constructor & Destructor Documentation

7.9.2.1 RoutingServiceProperty()

```
RoutingServiceProperty ( )
```

Constructor.

7.9.3 Member Data Documentation

7.9.3.1 cfgFile

```
String cfgFile
```

Path to an RTI Routing Service configuration file.

If not NULL, this file is loaded; otherwise, if cfg_strings is not NULL, that XML code is loaded

[default] NULL.

7.9.3.2 cfgStrings

```
String [] cfgStrings
```

XML configuration represented as strings.

An array of strings that altogether make up an XML document to configure RTI Routing Service. This parameter is used only if **cfgFile** (p. 39) is NULL.

For example:

```
property.cfg_strings = new String[3];
property.cfg_strings[0] = "<dds><routing_service>";
property.cfg_strings[1] = "<domain_route><participant><domai";
property.cfg_strings[2] = "n_id>0...</dds>";
```

The reason for using an array instead of one single string is to get around the limited size of literal strings in Java. In general, if you create the XML string dynamically the array needs only one element:

```
property.cfg_strings = new String[1];
// fill in property.cfg_strings[0]
```

[default] NULL.

7.9.3.3 serviceName

```
String serviceName
```

The name of the Routing Service instance to run.

This is the name used to find the <routing_service> XML tag in the configuration file; the name that will be used to refer to this execution in remote administration and monitoring.

[default] NULL (use RTI_RoutingService)

7.9.3.4 applicationName

```
String applicationName
```

Assigns a name to the execution of the RTI Routing Service.

Remote commands and status information will refer to the routing service using this name. In addition, the name of DomainParticipants created by RTI Routing Service will be based on this name.

[default] service_name if this value is different than NULL. Otherwise, "RTI_Routing_Service".

7.9.3.5 enforceXsdValidation

```
boolean enforceXsdValidation = true
```

Controls whether the service applies XSD validation to the loaded configuration.

[default] true

7.9.3.6 serviceVerbosity

```
int serviceVerbosity = RoutingService.LOG_VERBOSITY_EXCEPTIONS
```

The verbosity of the service.

Values:

- **RoutingService.LOG_VERBOSITY_SILENT** (p. 36)
- **RoutingService.LOG_VERBOSITY_EXCEPTIONS** (p. 36)
- **RoutingService.LOG_VERBOSITY_WARNINGS** (p. 36)
- **RoutingService.LOG_VERBOSITY_INFO** (p. 37)

[default] RoutingService.LOG_VERBOSITY_EXCEPTIONS (p. 36)

7.9.3.7 ddsVerbosity

```
int ddsVerbosity = RoutingService.LOG_VERBOSITY_EXCEPTIONS
```

The verbosity of the service.

Values:

- **RoutingService.LOG_VERBOSITY_SILENT** (p. 36)
- **RoutingService.LOG_VERBOSITY_EXCEPTIONS** (p. 36)
- **RoutingService.LOG_VERBOSITY_WARNINGS** (p. 36)
- **RoutingService.LOG_VERBOSITY_INFO** (p. 37)

[default] RoutingService.LOG_VERBOSITY_EXCEPTIONS (p. 36)

7.9.3.8 domainIdBase

```
int domainIdBase = 0
```

Value that is added to the domain IDs of the domain routes in the XML configuration.

By using this, an XML file can use relative domain IDs.

[default] 0

7.9.3.9 pluginSearchPath

```
String pluginSearchPath
```

This path is used to look for plug-in libraries.

If the plug-in class libraries specified in the XML file don't contain a full path, RTI Routing Service looks for them in this directory. If not present, it will rely on the system library path.

[default] NULL (current directory)

7.9.3.10 dontStartService

```
boolean dontStartService = false
```

Set this to true to if you do not want RTI Routing Service enabled when **RoutingService.start** (p. 34) is called.

RTI Routing Service can be enabled afterwards through remote administration.

[default] DDS_BOOLEAN_FALSE

7.9.3.11 enableAdministration

```
boolean enableAdministration = false
```

Set this to true to enable remote administration or false to disable it.

[default] DDS_BOOLEAN_FALSE

7.9.3.12 administrationDomainId

```
int administrationDomainId = 0
```

If **enableAdministration** (p. 41) is true, this is the domain ID to use for remote administration.

Takes precedence over the XML configuration. If **enableAdministration** (p. 41) is false, this value is not used even if remote administration is enabled in the XML configuration.

[default] 0

7.9.3.13 enableMonitoring

```
boolean enableMonitoring = false
```

Set it to true to enable remote monitoring or false to disable it.

[default] false

7.9.3.14 monitoringDomainId

```
int monitoringDomainId = 0
```

If **enableMonitoring** (p. 41) is true, this is the domain ID to use for remote monitoring.

Takes precedence over the XML configuration. If **enableMonitoring** (p. 41) is false, this value is not used, even if remote monitoring is enabled in the XML configuration.

[default] 0

7.9.3.15 skipDefaultFiles

```
boolean skipDefaultFiles = false
```

Set it to true to avoid loading the standard files usually loaded by RTI Routing Service.

Only the configuration in **cfgFile** (p. 39) or **cfgStrings** (p. 39) will be loaded.

[default] DDS_BOOLEAN_FALSE

7.9.3.16 identifyExecution

```
boolean identifyExecution = false
```

Set this to true to append the host name and process ID to the RTI Routing Service execution name.

Used to get unique names for remote administration and monitoring.

[default] false

7.9.3.17 licenseFileName

```
String licenseFileName
```

Path to an RTI Routing Service license file.

If not `NULL`, this file is checked for a valid license; otherwise, default location will be used. This parameter is not used in unlicensed versions.

[default] NULL.

7.9.3.18 user_environment

```
Properties user_environment = new Properties()
```

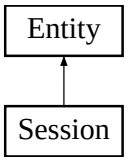
Dictionary of user variables. The dictionary provides a parallel way to expand XML configuration variables in the form , when they are not defined in the environment.

[default] empty

7.10 Session Interface Reference

Session (p. 43) interface (optional).

Inheritance diagram for Session:



Additional Inherited Members

7.10.1 Detailed Description

Session (p. 43) interface (optional).

A **Session** (p. 43) is a concurrency unit within a connection that has an associated set of StreamReaders and Stream↩ Writers. Access to the StreamReaders and StreamWriters in the same **Session** (p. 43) is serialized by RTI Routing Service.

In the XML configuration file, Sessions are associated with the tag <session> within a domain route. For each <session> tag, RTI Routing Service will create two adapter Sessions, one per connection.

7.11 StreamInfo Class Reference

Stream information.

Public Member Functions

- String **getStreamName** ()

  The stream name.
- **TypeInfo getTypeInfo** ()

  The type information associated with the stream.
- boolean **isDisposed** ()

  Indicates whether the stream is a newly discovered stream or a disposed stream that no longer exists.

7.11.1 Detailed Description

Stream information.

A stream in RTI Routing Service is a typed data channel to consume/produce data in one data domain.

7.11.2 Member Function Documentation

7.11.2.1 getStreamName()

```
String getStreamName ( )
```

The stream name.

7.11.2.2 getTypeInfo()

```
TypeInfo getTypeInfo ( )
```

The type information associated with the stream.

This field is invalid for disposed streams.

7.11.2.3 isDisposed()

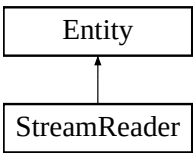
```
boolean isDisposed ( )
```

Indicates whether the stream is a newly discovered stream or a disposed stream that no longer exists.

7.12 StreamReader Interface Reference

StreamReader (p. 44) interface (optional).

Inheritance diagram for StreamReader:



Public Member Functions

- void **read** (List< Object > sampleList, List< Object > infoList) throws AdapterException

  Reads a collection of data samples and sample infos from an input stream.
- void **returnLoan** (List< Object > sampleList, List< Object > infoList) throws AdapterException

  Returns the loan on the read samples and infos.

7.12.1 Detailed Description

StreamReader (p. 44) interface (optional).

A **StreamReader** (p. 44) provides a way to read samples of a specific type from a data domain.

In the XML configuration file, StreamReaders are associated with the tag <input> within <route> and <auto_route>.

7.12.2 Member Function Documentation

7.12.2.1 read()

```
void read (
            List< Object > sampleList,
            List< Object > infoList ) throws  AdapterException
```

Reads a collection of data samples and sample infos from an input stream.

When RTI Routing Service is done using the samples, it will 'return the loan' to the **StreamReader** (p. 44) by calling **com.rti.routingservice.adapter.StreamReader.returnLoan** (p. 46).

Required: Only when the adapter is used to read data.

Parameters

sampleList	<<*out*>> List that will hold the output samples. The objects in the list are typically of the class com.rti.dds.dynamicdata.DynamicData (see the RTI Connext documentation). But in general, the data representation associated with the output samples will be given by the value of the connection attribute com.rti.routingservice.adapter.data_representation_kind that is obtained using the associated com.rti.routingservice.infrastructure.StreamInfo.
infoList	<<*out*>> List that will hold the output sample infos. It can be left empty if there is no info associated to the samples. The objects in the list are typically of the class com.rti.dds.subscription.SampleInfo (see the RTI Connext documentation). But in general the info representation associated with the output sample infos will be given by the value of the connection attribute com.rti.routingservice.adapter.info_representation_kind that is obtained using the associated com.rti.routingservice.infrastructure.StreamInfo.

Exceptions

AdapterException	- if an error occurs.

See also

com.rti.routingservice.adapter.StreamReader.returnLoan (p. 46)

7.12.2.2 returnLoan()

```
void returnLoan (
            List< Object > sampleList,
            List< Object > infoList ) throws  AdapterException
```

Returns the loan on the read samples and infos.

RTI Routing Service calls this method to indicate that it is done accessing the collection of data samples and sample infos obtained by an earlier invocation of **com.rti.routingservice.adapter.StreamReader.read** (p. 45).

Required: Only when the adapter is used to read data.

Parameters

sampleList	<<*in*>> List of samples.
infoList	<<*in*>> List of infos.

Exceptions

AdapterException	- if an error occurs.

See also

com.rti.routingservice.adapter.StreamReader.read (p. 45)

7.13 StreamReaderListener Interface Reference

StreamReaderListener (p. 46) used to notify Routing Service that new data is available.

Public Member Functions

- void **onDataAvailable** (**StreamReader** streamReader)

  Callback that notifies RTI Routing Service of new samples.

7.13.1 Detailed Description

StreamReaderListener (p. 46) used to notify Routing Service that new data is available.

RTI Routing Service uses the session threads (there is one per <session> tag) to read data from StreamReaders.

Each session thread will block waiting for new data using a WaitSet. When a **StreamReader** (p. 44) receives new data, it will use the **StreamReaderListener** (p. 46)'s **com.rti.routingservice.adapter.StreamReaderListener.on↩ DataAvailable** (p. 47) callback operation to wake up the associated session thread. After that, the session thread will invoke the **StreamReader** (p. 44)'s **com.rti.routingservice.adapter.StreamReader.read** (p. 45) operation to get the new data.

The following figure describes how the session thread reads samples from a **StreamReader** (p. 44).

7.13.2 Member Function Documentation

7.13.2.1 onDataAvailable()

```
void onDataAvailable (
              StreamReader streamReader )
```

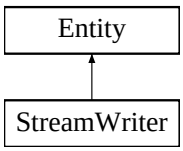
Callback that notifies RTI Routing Service of new samples.

When a **StreamReader** (p. 44) receives new data, it will use this callback to notify RTI Routing Service that there are new samples.

7.14 StreamWriter Interface Reference

StreamWriter (p. 47) interface (optional).

Inheritance diagram for StreamWriter:



Public Member Functions

- int **write** (List< Object > sampleList, List< Object > infoList) throws AdapterException

  Writes a collection of data samples to an output stream.

7.14.1 Detailed Description

StreamWriter (p. 47) interface (optional).

A **StreamWriter** (p. 47) provides a way to write samples of a specific type in a data domain.

In the XML configuration file, StreamWriters are associated with the tag <output> within <route> and <auto_route>.

7.14.2 Member Function Documentation

7.14.2.1 write()

```
int write (
            List< Object > sampleList,
            List< Object > infoList ) throws  AdapterException
```

Writes a collection of data samples to an output stream.

Required: Only when the adapter is used to write data.

Parameters

sampleList	<<*in*>> List of samples. The data representation associated with the samples will be given by the value of the connection attribute com.rti.routingservice.adapter.data_representation_kind that is obtained using the associated com.rti.routingservice.infrastructure.StreamInfo.
infoList	<<*in*>> List of samples info. The info representation associated with the sample infos will be given by the value of the connection attribute com.rti.routingservice.adapter.info_representation_kind that is obtained using the associated com.rti.routingservice.infrastructure.StreamInfo.

Returns

Number of samples written.

Exceptions

AdapterException	- if an error occurs.

7.15 TypeInfo Class Reference

Type information.

Public Member Functions

- **TypeInfo** (String typeName, Object typeRepresentation)

  TypeInfo *(p. 48) constructor.*
- String **getTypeName** ()

  Gets the registered type name.
- Object **getTypeRepresentation** ()

  Gets the type representation.

Static Public Attributes

- static int **DYNAMIC_TYPE_REPRESENTATION**

  Dynamic type representation.
- static int **XML_TYPE_REPRESENTATION**

  [Not supported] *XML type representation.*
- static int **JAVA_OBJECT_TYPE_REPRESENTATION**

  [Not supported] *Java object type representation.*
- static int **DYNAMIC_DATA_REPRESENTATION**

  Dynamic data representation.
- static int **XML_DATA_REPRESENTATION**

  [Not supported] *XML data representation.*
- static int **JAVA_OBJECT_DATA_REPRESENTATION**

  [Not supported] *Java object data representation.*

7.15.1 Detailed Description

Type information.

7.15.2 Constructor & Destructor Documentation

7.15.2.1 TypeInfo()

```
TypeInfo (
            String typeName,
            Object typeRepresentation )
```

TypeInfo (p. 48) constructor.

Parameters

typeName	rs_st_in Registered type name.
typeRepresentation	rs_st_in Type representation.

7.15.3 Member Function Documentation

7.15.3.1 getTypeName()

```
String getTypeName ( )
```

Gets the registered type name.

Returns

The registered type name.

7.15.3.2 getTypeRepresentation()

```
Object getTypeRepresentation ( )
```

Gets the type representation.

Returns

The type representation.

7.16 Version Class Reference

Represents the version number of an adapter plugin.

Public Member Functions

- **Version** (int major, int minor, int release, int revision)
  Creates the version of an adapter plugin.

7.16.1 Detailed Description

Represents the version number of an adapter plugin.

7.16.2 Constructor & Destructor Documentation

7.16.2.1 Version()

```
Version (
            int major,
            int minor,
            int release,
            int revision )
```

Creates the version of an adapter plugin.

Parameters

major	The major version number
minor	The minor version number
release	The release number
revision	The revision of a release

Chapter 8

File Documentation

8.1 RoutingService.java File Reference

RTI Routing Library API.

Classes

- class **RoutingService**
  RTI Routing Service.

8.1.1 Detailed Description

RTI Routing Library API.

8.2 RoutingServiceProperty.java File Reference

Properties for RTI Routing Service via API.

Classes

- class **RoutingServiceProperty**
  Configuration of RTI Routing Service.

8.2.1 Detailed Description

Properties for RTI Routing Service via API.

Index