RTI Connext

Core Libraries and Utilities

What's New

in Version 4.5f

Your systems. Working as one.

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

Trademarks

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

What's New

This document highlights new or changed features in *RTI® Connext™* (formerly *RTI Data Distribution Service*) 4.5f. (For details on fixed bugs, please see the *Release Notes*.)

For more information, visit the RTI Knowledge Base, accessible from https://supor.rti.com/, to see sample code, general information on *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 e-mail to **support@rti.com**. We can only guarantee a response to customers with a current maintenance contract or subscription. You can purchase a maintenance contract or subscription by contacting your local RTI representative (see http://www.rti.com/company/contact.html), sending an e-mail request to **sales@rti.com**, or calling +1 (408) 990-7400.

---

1 Introducing RTI Connext

RTI's family of products is now called *RTI Connext™*. *RTI Connext* is the first edge-to-enterprise real-time SOA platform. The products provide seamless, enterprise-wide integration to improve efficiency, responsiveness and integration with real-time business intelligence (BI).

With this release the *RTI Connext* family of products includes:

- *RTI Connext DDS* is the leading implementation of the Object Management Group (OMG) Data Distribution Service (DDS) specification. The proven technology is the foundation of the *RTI Connext* product family.
- *RTI Connext Messaging* provides a universal messaging infrastructure for ultra high-performance and real-time systems, including machine-to-machine (M2M)

and human-machine interface (HMI) applications. It includes APIs for JMS and DDS, *Persistence Service* for late joiners, *Recording Service* for logging data for deep analysis and archiving, *Federation Service* for securely bridging applications and systems across both local and wide area networks, and tools for monitoring, analyzing and debugging your complete system.

- *RTI Connext Integrator* delivers rapid integration of operational systems with enterprise-wide IT without requiring modification to those systems. *Connext Integrator* is a service bus, similar to an enterprise service bus (ESB), but designed for the demanding real-time environment. It provides the backbone of a real-time Service Oriented Architecture (SOA).

2 New Architectures

Connext 4.5f adds support for the following new architectures:

SUSE Linux Enterprise 11 Server Service Pack 1 (2.6 kernel)	AMD64	gcc 4.3.4	x64Linux2.6gcc4.3.4
		Sun Java Platform Standard Edition JDK 1.6	x64Linux2.6gcc4.3.4jdk
RedHawk Linux 6.0[1]	64-bit	gcc 4.4.5	x64Linux2.6gcc4.4.5
LynxOS 4.0	PPC 604 PPC 7XX (such as 750)	gcc 3.2.2	ppc750Lynx4.0.0gcc3.2.2
		Sun Java Platform Standard Edition JDK 1.4	ppc750Lynx4.0.0gcc3.2.2jdk

1. This platform is only available as a Custom Target Library (CTL). Contact **sales@rti.com** for details. It uses the same target libraries as Red Hat Enterprise Linux 6.1.

3 Experimental Features

This release introduces the concept of *experimental features*, which are used to evaluate potential new features and obtain customer feedback. They are not guaranteed to be consistent or supported and they should not be used in production.

Experimental features are clearly documented as such in the *Core Libraries and Utilities What's New* document or the *Release Notes* document of the component in which they are included, as well as in the component's *User's Manual*.

For more information on the concept of experimental features, see the *RTI Core Libraries and Utilities Release Notes*.

4 Experimental Feature: XML-Based Application Creation

In this release, you can evaluate an experimental feature (see Section 3) known as XML-Based Application Creation. This feature simplifies the development and programming of *RTI Connext* applications by allowing you to define your *entire* system using XML.

With previous releases, you could define data types and Quality of Service settings in XML. With this feature, you can also use XML to define your system's *Topics*, *DomainParticipants*, and all the *Entities* they contain (*Publishers*, *Subscribers*, *DataWriters* and *DataReaders*).

For details on using this feature, see the *XML-Based Application Creation Getting Started Guide (***RTI_Connext_XML_AppCreation_GettingStarted.pdf***)*.

Note: Experimental features cannot be used in a production system. See the *Release Notes* for more information.

5 Change to Default Value for UDPv4 Transport Property reuse_multicast_receive_resource

The default value for the UDPv4 transport property, **reuse_multicast_receive_resource**, has been changed from disabled to enabled. This change prevents Linux applications running with the Linux firewall enabled from hanging during *DomainParticipant* shutdown.

[RTI Bug # 14221]

6 New UDPv4 Transport Property to Set Protocol Overhead

In the UDPv4 transport, the existing property **message_size_max** does not account for UDPv4 and IP headers. As a result, sizing **message_size_max** too close to the UDPv4 datagram size limit of 65535 bytes will result in messages too large for the *DataWriter* to send. A new UDPv4 transport property, **protocol_overhead_max**, accounts for the pro-

tocol overhead, as well as any other protocol-related overhead; it is set by default to account for UDPv4 and IPv4 headers. Furthermore, the middleware will automatically limit the effective **message_size_max** to be no larger than the UDPv4 limit (65535) minus this overhead.

7 More Flexibility in Allow/Deny Interface Specification for UDPv6 transport

This release provides more flexibility in terms of filter specification for allowing or denying interfaces for the UDPv6 transport. The new method is case-insensitive and accepts addresses and patterns in IPv6 notation. It also supports the wildcard character, *, which can expand up to 4 digits in a block. The wildcard must be either leading or trailing. Multiple wildcards can be specified in a single filter, but only one wildcard can be specified per block (between colons).

Examples of valid filters:

Filter	Same as	Matches
::*:*:*:*:*:*		Any IPv6 interface
FE80::*:*	fe80::*:*	FE80:0000:0000:0000:0000:0000:xxxx:xxxx
	Fe80:0:0::*:*	
	Fe80:0:0:0:0:0:*:*	
FE80:aBC::202:2*:*:*2		FE80:0ABC:0000:0000:0202:2xxx:xxxx:xxx2
eth*		eth0, eth1, etc.

[RTI Bug # 14075]

8 Ability to Configure EntityFactoryQosPolicy in DomainParticipantFactory using XML

This release adds the ability to configure the EntityFactoryQosPolicy in the DomainParticipantFactory using XML. This is done using a new XML tag called <entity_factory> under <participant_factory_qos>. For example:

```
<participant_factory_qos>
  <entity_factory>
```

```
    <autoenable_created_entities>false</autoenable_created_entities>
  </entity_factory>
</participant_factory_qos>
```

9 C++ Code Generation Enables Easier C++ Template Programming

The C++ type definition for structs, valuetypes, and unions has been enhanced with traits and definitions for *DataWriter*, *DataReader*, TypeSupport, and Sequence in order to make template programming easier.

For example, the Foo IDL type below generates the following C++ type definition:

IDL:

```
struct Foo {
    short myShort;
};
```

C++:

```
typedef struct Foo
{
  #ifdef __cplusplus
    typedef struct FooSeq Seq;
  #ifndef NDDS_STANDALONE_TYPE
    typedef FooTypeSupport TypeSupport;
    typedef FooDataWriter DataWriter;
    typedef FooDataReader DataReader;
  #endif
  #endif
    DDS_Short  myShort;
} Foo;
```

The above type definition will make it easier for you to create class templates using Foo as the type parameter.

For example (note: error checking omitted for the sake of brevity):

```
template <typename T>
void take_and_print(typename T::DataReader* reader)
{
    DDS_SampleInfo info;
    T * sample = T::TypeSupport::create_data();
    DDS_ReturnCode_t result = reader->take_next_sample(*sample,
                                                       info);
```

```
    if (result == DDS_RETCODE_OK && info.valid_data) {
        T::TypeSupport::print_data(sample);
    }
    T::TypeSupport::delete_data(sample);
}

void main(int argc, char ** argv)
{
    // ...
    FooDataReader foo_dr = ...;
    BarDataReader bar_dr = ...;
    // ...
    take_and_print<Foo>(foo_dr);
    take_and_print<Bar>(bar_dr);
}
```

10 Interoperability Between 64-bit and 32-bit Connext Applications using Shared Memory

Previous releases already supported interoperability with the release core libraries. However, applications using the debug libraries failed with the following error message:

```
REDAConcurrentQueue_attach:!precondition: !((((RTI_UINT64)(memAd-
dress)) % (sizeof(struct REDAConcurrentQueueStructToDetermineAlign-
ment))) == 0)
NDDS_Transport_Shmem_attach_writer:failed to initialize unable to
attach
```

This release adds interoperability over shared memory using the debug core libraries.

11 Support for Circular Dependencies on Included Files in XML Type Declarations

In previous releases, the following XML type declarations were not allowed since there were circular dependencies between the included files in types1.xml and types2.xml:

types1.xml

```
<types>
    <include file="types2.xml"/>

    <struct name="MyPrimitiveStruct" topLevel="true">
        <member name="m1" type="boolean"/>
        <member name="m2" type="octet"/>
        <member name="m3" type="char"/>
        <member name="m4" type="nonBasic" nonBasicType-
Name="MyPrimitiveStruct2"/>
    </struct>
</types>
```

types2.xml

```
<types>
    <include file="types1.xml"/> <!-- This was not allowed -->

    <struct name="MyPrimitiveStruct2" topLevel="true">
        <member name="m1" type="boolean"/>
        <member name="m2" type="octet"/>
        <member name="m3" type="char"/>
    </struct>
</types>
```

This problem has been fixed in this release and circular dependencies on included files are allowed. Notice that circular dependencies on included files are not the same as circular dependencies on types. For example, the following XML is still not allowed:

types1.xml

```
<types>
    <include file="types2.xml"/>

    <struct name="MyPrimitiveStruct" topLevel="true">
        <member name="m1" type="boolean"/>
        <member name="m2" type="octet"/>
        <member name="m3" type="char"/>
        <member name="m4" type="nonBasic"
                nonBasicTypeName="MyPrimitiveStruct2"/>
    </struct>
</types>
```

types2.xml

```
<types>
    <include file="types1.xml"/> <!-- This is allowed -->

    <struct name="MyPrimitiveStruct2" topLevel="true">
        <member name="m1" type="boolean"/>
        <member name="m2" type="octet"/>
        <member name="m3" type="char"/>
        <member name="m4" type="nonBasic"
         nonBasicTypeName="MyPrimitiveStruct"/> <!-- Not allowed -->
    </struct>
</types>
```