RTI Code Generator

for

RTI Connext DDS

Release Notes

Version 2.3.0

Your systems. Working as one.

© 2015 Real-Time Innovations, Inc.
All rights reserved.
Printed in U.S.A. First printing.
June 2015.

Trademarks

Real-Time Innovations, RTI, NDDS, RTI Data Distribution Service, DataBus, Connext, Micro DDS, the RTI logo, 1RTI and the phrase, "Your Systems. Working as one," are registered trademarks, trademarks or service marks of Real-Time Innovations, Inc. All other trademarks belong to their respective owners.

Copy and Use Restrictions

No part of this publication may be reproduced, stored in a retrieval system, or transmitted in any form (including electronic, mechanical, photocopy, and facsimile) without the prior written permission of Real-Time Innovations, Inc. The software described in this document is furnished under and subject to the RTI software license agreement. The software may be used or copied only under the terms of the license agreement.

Technical Support

Real-Time Innovations, Inc.
232 East Java Drive
Sunnyvale, CA 94089

Phone:	(408) 990-7444
Email:	support@rti.com
Website:	https://support.rti.com/

Contents

Release Notes

1 Supported Platforms

You can run *RTI® Code Generator* as a Java application or, for performance reasons, as a native application that invokes Java. See the *RTI Code Generator User's Manual*.

As a Java application, *Code Generator* is supported on all host platforms (listed in the *RTI Connext DDS Core Libraries Release Notes*[1]) by using the script *rtiddsgen*.

As a native application, *Code Generator* is supported on the following platforms by using the script *rtiddsgen_server*:

- ❑ CentOS 6.0, 6.2, 6.3, 6.4 (gcc 4.4.5)
- ❑ Red Hat® Enterprise Linux 5.0 (gcc 4.1.1)
- ❑ Red Hat Enterprise Linux 6.0, 6.1, 6.2, 6.3, 6.4 (gcc 4.4.5)
- ❑ Windows® 7
- ❑ Windows 8
- ❑ Windows Server 2003
- ❑ Windows Server 2008 R2
- ❑ Windows Server 2012 R2
- ❑ Windows Vista®
- ❑ Windows XP Professional

For details on these platforms, see the *RTI Connext DDS Core Libraries Release Notes*.

2 What's New in 2.3.0

2.1 Performance and Usability Improvements

This implementation of *Code Generator* significantly improves the performance of the original implementation and makes it easier to customize the generated output.

1. This document is available from the RTI Community Portal's Documentation page.

2.2 Option to Configure Name of Macro for Exporting Symbols when Building Windows DLL

This release introduces a new command-line option, **-dllExportMacroSuffix**, which allows you to configure the suffix of the macro used to export type-plugin symbols when building a Windows DLL.This option works for C, C+, C+/CLI and .NET languages.

If you run *rtiddsgen* without this option, the macro is named **NDDS_USER_DLL_EXPORT**. With this option, the macro is named **NDDS_USER_DLL_EXPORT_*suffix***, where *suffix* is provided after the option (**-dllExportMacroSuffix suffix**).

2.3 Support for Enumerators with Duplicate Values

Although the Extensible Types specification does not support enumerators with duplicate values, *Code Generator* now generates compatible code with them in C, C++, .NET, and Java. Please note that unions based on an enumerator with duplicate values are not supported.

2.4 Project Files for Java Examples

Code Generator can now generate an Ant file (**build.xml**) and an Eclipse project for Java examples (in addition to the makefile generated in previous releases).

2.5 New -express Flag for Compatibility with Microsoft® Visual Studio® Express 2008 and 2010

In the previous release, the default project files generated for a C# example could not be built with Microsoft Visual Studio Express versions 2008 and 2010.

Now you can generate project files that can be built with Microsoft Visual Studio Express 2008 and 2010 by using the new **-express** flag.

With this flag, *Code Generator* will create two solutions:

- ❑ **<fileName>_type-dotnet<version>.sln** — Build this first with the C++ Microsoft Visual Studio Express
- ❑ **<Foo>_example-chsarp.sln** — Build this one after the previous one, with C# Microsoft Visual Studio Express

Note: The **-express** flag is only compatible with i86Win32VS2008 and i86Win32VS2010 architectures; newer versions of Microsoft Visual Studio Express do not need this flag.

2.6 Support for Unbounded Sequences and Strings in .NET, C, and C++ Code Generation

In previous releases, RTI assigned a default bound to sequences and strings of unspecified bound. The default bound for sequences is 100 elements; the default bound for strings is 255 characters. You can override these default values by using *Code Generator's* command-line options, **-sequenceSize** and **-stringSize**, respectively.

To support unbounded sequences and strings, *Code Generator* has a new command-line option: **-unboundedSupport**. This new option may only be used when generating code for .NET, C, and C++ (that is, the **-language** option must be specified for C++/CLI, C#, C, or C++).

For sequences: The generated code will deserialize incoming samples by dynamically allocating and deallocating memory to accommodate the actual size of sequences. Specifically, if a sequence is being received into a sample from the *DataReader's* cache, the old memory for the sequence will be deallocated and memory of sufficient size to hold the deserialized data will be allocated. When initially constructed, sequences will not pre-allocate any elements, thus having a maximum of zero elements. Dynamic memory allocation will be applied only to unbounded sequences.

To use the command-line option **-unboundedSupport**, you must also use the threshold QoS properties **dds.data_writer.history.memory_manager.fast_pool.pool_buffer_max_size** on the *DataWriter* and **dds.data_reader.history.memory_manager.fast_pool.pool_buffer_max_size** on the *DataReader.* In addition, the QoS value **reader_resource_limits.dynamically_allocate_fragmented_samples** on the *DataReader* must be set to true.

Example XML file:

```
<qos_profile name="Unbounded_Profile">
    <datawriter_qos>
        <property>
            <value>
                <element>
                    <name>
     dds.data_writer.history.memory_manager.fast_pool.pool_buffer_max_size
                    </name>
                    <value>4096</value>
                  </element>
            </value>
        </property>
    </datawriter_qos>

    <datareader_qos>
        <reader_resource_limits>
            <dynamically_allocate_fragmented_samples>
                true
            </dynamically_allocate_fragmented_samples>
        </reader_resource_limits>
        <property>
            <value>
                <element>
                    <name>
     dds.data_reader.history.memory_manager.fast_pool.pool_buffer_max_size
                    </name>
                    <value>4096</value>
                  </element>
            </value>
        </property>
    </datareader_qos>
</qos_profile>
```

For additional information on these QoS values, see the *RTI Connext DDS Core Libraries User's Manual*.

2.7 C++ Code no Longer Generated when Converting to XML or IDL

In previous releases when *Code Generator* was used with the options **-convertToXML** or **-convertToIDL**, it generated C++ files in addition to the requested XML or IDL file. In this release, C++ files will be not generated when you use the options **-convertToXML** or **-convertToIDL**.

2.8 Support for Optional Members in .NET

This release adds support for optional members in .NET, as defined in the "Extensible and Dynamic Topic Types for DDS" (DDS-XTypes) specification from the Object Management Group (OMG). Optional members were already supported in C, C++ and Java.

In a structure type, an optional member is a member that an application can decide to send or omit as part of every published sample. Specifically, these features are now supported:

- ❏ Declaring struct members as optional in IDL and XML
- ❏ Generating code for types with optional members in C, C++, C#, and Java
- ❏ Accessing and setting optional members in the existing DynamicData API using the member name. Accessing or setting by member ID is not supported at this time.
- ❏ Using content filters for types with optional members

For more information on using optional members, see the updated *RTI Connext DDS Core Libraries Getting Started Guide Addendum for Extensible Types.*

2.9 TypeSupport Operations to Serialize Sample into Buffer and Deserialize Sample from Buffer

This release provides two new TypeSupport operations to serialize a sample into a buffer and deserialize a sample from a buffer. The sample serialization/deserialization uses CDR representation.

The feature is supported in the following languages: C, C++, Java, and .NET.

C:

```
#include "FooSupport.h"
FooTypeSupport_serialize_data_to_cdr_buffer(...)
FooTypeSupport_deserialize_data_from_cdr_buffer(...)
```

C++:

```
#include "FooSupport.h"
FooTypeSupport::serialize_data_to_cdr_buffer(...)
FooTypeSupport::deserialize_data_from_cdr_buffer(...)
```

Java:

```
FooTypeSupport.get_instance().serialize_to_cdr_buffer(...)
FooTypeSupport.get_instance().deserialize_from_cdr_buffer(...)
```

C++/CLI:

```
FooTypeSupport::serialize_data_to_cdr_buffer(...)
FooTypeSupport::deserialize_data_from_cdr_buffer(...)
```

C#:

```
FooTypeSupport.serialize_data_to_cdr_buffer(...)
FooTypeSupport.deserialize_data_from_cdr_buffer(...)
```

2.10 Functionality Removed from Previous Release

The following command-line options have been removed:

- ❏ **-convertToCcl, -convertToCcs, -convertToWsdl, -convertToXsd**
- ❏ **-debug**
- ❏ **-expandCharSeq, -expandOctetSeq**
- ❏ **-inputWsdl, -inputXsd**
- ❏ **-optimization**

The following IDL type is not supported:

- ❏ bitfields

3 What's Fixed in 2.3.0

3.1 Compilation Warning in C/C++ Generated Code for Typedefs

When generating code for a typedef type, users compiling with the **-Wunused-variable** (or **-W**) option may have seen this warning message:

```
warning: unused variable deallocParams
```

This warning is now avoided.

[RTI Issue ID CODEGEN-665]

3.2 Incorrect Java Code Generated when Top Module of Enum Type was 'i'

Incorrect Java code was generated if the IDL contained an enum type inside a top module named '**i**'. For example:

```
module i {
    enum MyEnum{
        e1,
        e2
    };
    struct MyStruct {
        MyEnum my_enum;
    };
};
```

The generated Java code did not compile and reported this error:

```
sm[i]=new StructMember("my_enum", false, (short)-1, false,(TypeCode) i.MyE-
num.VALUE,0, false);i++;

^ error int cannot be dereferenced
```

This problem has been resolved.

[RTI Issue ID CODEGENII-152]

3.3 Memory Leak when Finalizing Pointer to Typedef Pointer of Strings in C

If a data type in IDL included a pointer to a typedef of a string pointer, like in this example:

```
typedef string<100> * StringPointer;
struct PointerStruct{
   StringPointer * ppStrData;
};
```

The C code generated for the finalize method of the type containing that member (PointerStruct in the example) was incorrect and caused a memory leak. This issue has been resolved.

[RTI Issue ID CODEGENII-154]

3.4 Incorrect Suffix in Generated Code for 'long long' and 'unsigned long long' Constant Definitions in C/C++

When generating code for a 'long long' or an 'unsigned long long' constant in C, the generated code was missing the corresponding suffix in the value (LL or ULL, respectively). In the case of C++, the generated code was missing the suffix for 'unsigned long long' (ULL). This issue has been resolved and the constants are defined as in the following example:

Constants in C:

```
#define LONG_LONG_CONST (2147483648LL)
#define ULONG_LONG_CONST (2147483648ULL)
```

Constant in C++:

```
static const DDS_LongLong LONG_LONG_CONST= 2147483648LL;
static const DDS_UnsignedLongLong ULONG_LONG_CONST= 2147483648ULL;
```

[RTI Issue ID CODEGENII-157]

3.5 Generated C++ Code had Invalid References for Unions with '@top-level false'

When generating C++ code for a Foo union type defined with **//@ top-level false**, the following invalid typedef references were included in the generated **Foo.h** file:

```
#ifndef NDDS_STANDALONE_TYPE
typedef FooTypeSupport TypeSupport;
typedef FooDataWriter DataWriter;
typedef FooDataReader DataReader;
#endif
```

The above lines caused compilation errors. This problem has been resolved.

[RTI Issue ID CODEGENII-158]

3.6 Directive '@resolve-name false' not Applied to Base Type when used in Derived Type

The directive '//@resolve-name false' was not applied correctly to a base type when used in the derived type. For example:

```
struct A: B {
    C m1;
}; //@resolve-name false
```

Consequently, the generated code for B was wrong, although it may have compiled. This problem has been resolved.

[RTI Issue ID CODEGENII-206]

3.7 Invalid Java Code Generated for Types Containing Primitive Optional Members

In previous releases, Java code generated for an IDL type containing optional members with any the following types primitive types did not compile:

- ❏ boolean
- ❏ long long
- ❏ unsigned long long
- ❏ float
- ❏ double
- ❏ long double

For example:

```
struct MyType {
    double m1; //@Optional
};
```

This problem has been resolved.

[RTI Issue ID CODEGENII-182, CODEGEN-646]

3.8 Finalize Methods with NULL Samples may have Caused Segmentation Fault—C/C++ APIs Only

The generated methods to finalize samples in C/C++ did not check to see if the sample was NULL. If the sample was NULL, this may have caused a segmentation fault. The problem has been resolved.

[RTI Issue ID CODEGENII-185]

3.9 Java NullPointerException when using Foo.copy_from() on Data Type with Optional Members

The **copy_from()** method generated for an IDL type in Java may have thrown a NullPointerException if the type contained optional members. For example:

```
struct InnerStr{
    long m1;
};
struct OuterStr{
    InnerStr opt_m1; //@Optional
};
```

In the above example, invoking **copy_from()** on an OuterStr object, **dst**, would cause a NullPointerException if the **opt_m1** member was set to null in the other OuterStr object, **src**.

This problem has been resolved

[RTI Issue IDs CODEGENII-190, CODEGEN-655]

3.10 Possible Compilation Error for Constant Octets in Java

The value of a constant octet in a Java declaration should have been cast to a byte but it was not. The generated Java code did not compile, for example, if the value was a hexadecimal value. This problem has been resolved.

[RTI Issue ID CODEGENII-191]

3.11 Incorrect Code Generated when '-namespace' used with '-language C'

The option **-namespace** is intended to be used only with **-language C++**. In the previous release, specifying both **-namespace** and **-language C** resulted in incorrect code. The problem has been resolved; -namespace will be ignored when used with -language C.

[RTI Issue ID CODEGENII-192]

3.12 Creating/Updating Examples or Makefiles Depended on Also Creating/Updating Type Files

If you used the -create/update <exampleFiles|makefiles> options without also specifying **-create/update typeFiles**, this caused the following error:

```
"ERROR com.rti.ndds.nddsgen.emitters.CSourceEmitter The last top-level type
variables weren't initialized. Example files wouldn't be generated"
```

The expected files were not generated.

In addition, trying to create/update a makefile resulted in the creation/update of example files instead of makefiles.

These problems have been resolved.

[RTI Issue ID CODEGENII-193]

3.13 Generated Ada Code for IDL with "include" Directive did not Compile

The Ada generated code for an IDL file with the "include" directive did not compile. For example, suppose the file **A.idl** contained this:

```
#include "B.idl";

struct myStruct{
    includedStruct m1;
};
```

And **B.idl** contained:

```
struct includedStruct {
    long m2;
}
```

The generated Ada code incorrectly specified **A_IDL_File** as the package for **includedStruct** while its package should be **B_IDL_File**.

It was also missing the corresponding with clause to include the **B_IDL_File** package.

These problems have been resolved.

[RTI Issue ID CODEGENII-195]

3.14 C++ Examples Generated with -namespace Option did not Compile if IDL Contained Modules

When generating a C++ example using the **-namespac**e option and an IDL data type that contained modules, the generated example did not compile. Variables inside a namespace were not properly generated in the publisher and subscriber code. This problem has been resolved.

[RTI Issue ID CODEGENII-200]

3.15 Generated Code for Struct with Keys and Copy Directive did not Compile

The generated code for a struct with key members and a **@copy** directive did not compile. For example:

```
struct MyStruct{
    //@copy  /*Information about foo*/
    short foo; //@key
};
```

Specifically, the generated C, C++, C#, or Java code was wrong and did not compile. This problem has been resolved.

[RTI Issue ID CODEGENII-204]

3.16 Getting Default Member of a Union may have Caused Error—Java API Only

For a union type with a default member that had a case label, such as this one:

```
union UnionType switch(short) {
    case 1: short m1;
    case 2: float m2;
    case 3:
    default: long m3;
};
```

The generated Java code may have thrown an error if the default discriminator was set to a case value that shared the default member (3 in the above example) and you tried to get that default member (m3 in the above example). This problem has been resolved.

[RTI Issue ID CODEGENII-205]

3.17 Memory Leak when Finalizing Array of Pointers of Non-Basic Types in C/C++

If a data type in IDL included an array of pointers of non-basic types, the generated C/C++ code may have caused a memory leak. For example:

```
struct PrimitiveType{
    long m1;
}
struct ArrayOfPointers{
    PrimitiveType * ptrMember [2];
}
```

The generated C/C++ code for the finalize method of the type containing that member (**ptrMember** in the above example) was incorrect and caused a memory leak. This problem has been resolved.

[RTI Issue ID CODEGENII-213]

3.18 Generated Ada Code for IDL with Names in Uppercase may not have Compiled

When not using modules, the generated Ada filenames used the IDL filename as a prefix. If the IDL filename contained uppercase letters, this caused a compilation error for compilers in which the default setting is to use lowercase filenames.

The problem has been resolved; now all generated Ada filenames are lowercase.

[RTI Issue ID CODEGENII-222]

3.19 Java Code Generated for Mutable Unions with 'Fall-through' Case Statements did not Compile

The Java code generated for a union with mutable extensibility that contained a fall-through case did not compile. An example of this kind of union is the following:

```
union MyUnionLongMutable switch (long){
        case 0:
        case 1:
            long m1;
        case 2:
            long m2;
    }; //@Extensibility MUTABLE_EXTENSIBILITY
```

This problem has been resolved.

[RTI Issue ID CODEGENII-235]

3.20 TypeSupport Operations to Get TypeCode

The previous release was missing TypeSupport operations to get the TypeCode. The feature is supported now in the following languages: C, C++, Java, and .NET.

C:

```
#include "FooSupport.h"
FooTypeSupport_get_typecode()
```

C++

```
#include "FooSupport.h"
FooTypeSupport::get_typecode()
```

Java:

```
FooTypeSupport.getTypeCode()
```

C++/CLI:

```
FooTypeSupport::get_typecode()
```

C#:

```
FooTypeSupport.get_typecode()
```

This feature is also supported for the Built-in Types. For example, for the Octets built-in type the operations are:

C:

```
DDS_OctetsTypeSupport_get_typecode()
```

C++

```
DDS::OctetsTypeSupport::get_typecode()
```

Java:

```
import com.rti.dds.type.builtin.BytesTypeSupport;
BytesTypeSupport.getTypeCode()
```

C++/CLI:

```
DDS::BytesTypeSupport::get_typecode()
```

C#:

```
using DDS;
BytesTypeSupport.get_typecode()
```

[RTI Issue ID CODEGENII-245, CODEGEN-540]

3.21 Java Makefile did not Compile if IDL Contained Modules

The generated makefiles for Java examples were incorrect if the IDL contained modules. The compilation rules for the example did not work correctly and the *Publisher/Subscriber* were not compiled. This problem has been resolved.

[RTI Issue ID CODEGENII-269]

3.22 Missing 'resolveName' Directive when Converting from XML to IDL

When converting an XML file with a type that specified the 'resolveName' directive, the generated IDL file was missing the corresponding 'resolveName' annotation. For example:

```
<struct name="MyStruct" resolveName="false">
    <member name="m1" type="nonBasic" nonBasicTypeName="MyStruct2" />
</struct>
```

This issue has been resolved.

[RTI Issue ID CODEGENII-270]

3.23 Duplicate Variable Names in IDL File not Reported as Error

Code Generator generated code for an IDL type containing duplicate member names without reporting any error. For example:

```
struct MyTestStruct
{
    octet myOctet_;
    octet myOctet_; // PROBLEM #1: duplicate field name
    };
```

The generated code wouldn't compile. This problem has been resolved. Now *Code Generator* will report an error and won't generate any code for that kind of IDL type.

[RTI Issue ID CODEGENII-275, CODEGEN-324]

3.24 C# Example did not Compile if IDL File Contained Modules

The generated C# example code for an IDL file that contained modules did not compile. The example was missing the corresponding namespace definition for the modules. This problem has been resolved.

[RTI Issue ID CODEGENII-309]

3.25 Non-Mutable Types with Keys and Optional Members did not Compile

The generated C/C++ code for non-mutable types (such as final or extensible) that contained both key and optional members did not compile. This problem has been resolved.

[RTI Issue ID CODEGENII-375]

3.26 Unable to Generate Code for Derived Valuetype with No Elements

Code Generator reported an error when trying to generate code for a derived valuetype that did not contain elements. No code was generated. This problem has been resolved.

[RTI Issue ID CODEGENII-378]

3.27 Generated Code for Types with Copy Directive did not Compile in C#

The generated code in C# for a type that contained an **@copy** directive did not compile. This problem has been resolved.

[RTI Issue ID CODEGENII-382]

3.28 Wrong TYPENAME Definition for a Type within a Module in C# and C++/CLI

The generated TYPENAME definition in the C# or C++/CLI code for a type within a module was missing the namespace prefix corresponding to the module. This problem has been resolved.

[RTI Issue ID CODEGENII-383]

3.29 #include not Processed for Unions in Ada

If an IDL type contained a **#include** directive, but the elements in the included IDL were only referenced within a union, the **#include** directive was not correctly processed and no **with** clause was added to the Ada code. The resulting code failed to compile. This problem has been resolved.

[RTI Issue ID CODEGENII-384]

3.30 Possible Poor Performance when Generating Code for Files with Many Modules

If the IDL file being compiled contained a lot of modules, including modules that are reopened multiple times, code generation may have been slow (regardless of whether or not you use fully qualified names for the type references within the IDL file). In this release, the problem has been resolved for the case in which you primarily use fully qualified names to refer to types or constant or enumerator.

As a best practice, RTI recommends that you use fully qualified names to refer to types.

For example, you should use a fully qualified name for a C type, such as this:

```
module A{
   module B{
       struct C {
          long m1;
       };
       struct D{
          ::A::B::C m2;
       };
   };
};
```

Use the above, instead of using a relative name for the C type like this:

```
module A{
   module B{
      struct C {
         long m1;
      };
      struct D{
         C m2;
      };
   };
};
```

[RTI Issue ID CODEGENII-387]

3.31 Duplicate Constants in IDL not Detected

Code Generator did not detect if there were two defined constants with the same name in an IDL type and it generated code without showing an error. The generated code would not compile. This problem has been resolved.

[RTI Issue ID CODEGENII-389]

3.32 Interoperability Issue with Enums with Unordered Indexes

In previous releases of *Code Generator*, the type-code generated in Java for an enum with unordered indexes was reordering them. Because this reordering did not occur for other languages, this caused interoperability problems between a Java application and a non-Java application using an enum with unordered indexes.

This problem has been resolved. In this release of *Code Generator*, the Java type-code for an enum with unordered indexes will no longer be reordered.

[RTI Issue ID CODEGENII-397]

3.33 Struct Inheritance was not Supported in Ada

When trying to generate Ada code for a struct that inherits from another struct, *Code Generator* threw the following error and didn't generate code.

```
ERROR com.rti.ndds.nddsgen.Main Fail:  com.rti.ndds.nddsgen.antlr.Idl-
StructTree cannot be cast to com.rti.ndds.nddsgen.antlr.IdlValueTree
```

This problem has been resolved.

[RTI Issue ID CODEGENII-399]

3.34 Unified Topic Name in Generated Examples Across Languages

In previous versions of *Code Generator*, the topic name for types within a module was not the same in all the languages. For example, consider this IDL:

```
module myModule {
    struct Hello{
      long m1;
    };
  };
```

The generated topic name in C and C++ was "Example myModule_Hello", while in Java, .NET, and Ada, it was "Example Hello".

This caused interoperability problems in communication between applications of different languages. This problem has been resolved. Now the generated topic in all languages is the fully qualified name with underscore, in the example, "Example myModule_Hello".

[RTI Issue ID CODEGENII-403]

3.35 Incorrect Code Generated for Type with Top-Level Directive with No Value

For an IDL type in which the **top-level** directive had no value, *Code Generator* incorrectly assumed that meant 'top-level false'. For example:

```
struct Foo {
    short myShort;
};//@top-level
```

Therefore, *Code Generator* did not generate any DataWriter or DataReader methods.

This issue has been resolved. For the above example, *Code Generator* will generate all the DataWriter and DataReader methods as if the type was declared with a //@top-level true directive.

[RTI Issue ID CODEGENII-410]

3.36 Incorrect C and Ada Code Generated for IDL Containing Forward Declarations

The generated C and Ada code for an IDL type that contained a forward declaration was incorrect and did not compile.

For example, in C:

```
struct MyStruct

struct MyStruct2 {
    MyStruct m1; //@Optional
};

struct MyStruct {
    long m1;
};
```

For Ada, the generated code was missing the type forward declaration in the corresponding specification file.

This problem has been resolved.

[RTI Issue ID CODEGENII-415/417]

3.37 Incorrect Code Generated for Union Forward Declaration

Code Generator was generating code in C, C++, and .NET for a union forward declaration as if it was an actual union declaration. This caused duplicated code when the actual union code was generated and the code would not compile. This problem has been resolved.

[RTI Issue ID CODEGENII-416]

3.38 Incorrect Code Generated for Mutable Struct that Inherited from Struct with Keys

For IDL containing a mutable struct inheriting from a keyed struct, like in this example:

```
struct MutableStruct : BaseStruct {
  float m2;
}; //@Extensibility MUTABLE_EXTENSIBILITY

struct BaseStruct{
  string<128> color; //@Key
  long x;
  long y;
}; //@Extensibility MUTABLE_EXTENSIBILITY
```

The generated code in C, C++, .Net, and Java was incorrect and did not compile. This problem has been resolved.

[RTI Issue ID CODEGENII-433]

3.39 Incorrect Code Generated when Multiple Annotations on Same Line

Code Generator does not support multiple annotations on the same line in an IDL struct, like in this example:

```
struct Shape1MutableExplicitID {
  string<STR_LEN_MAX> color; //@Key //@ID 10
  long x; //@ID 20
  long y;
  long shapesize; //@ID 30
}; //@Extensibility MUTABLE_EXTENSIBILITY
```

This generated incorrect code because it only used the first annotation on the line.

Now *Code Generator* will show a warning when it finds multiple annotation on the same line and it will not generate code.

If you need multiple annotations, write them on separate lines, like in this example:

```
struct Shape1MutableExplicitID {
  string<STR_LEN_MAX> color; //@Key
                             //@ID 10
  long x; //@ID 20
  long y;
  long shapesize; //@ID 30
}; //@Extensibility MUTABLE_EXTENSIBILITY
```

[RTI Issue ID CODEGENII-434]

3.40 Unexpected Warning in print_data Method for Type with Sequence Member

When compiling the generated C or C++ code for IDL with a type containing a sequence member, you may have seen the following warning:

```
warning: comparison of address of 'sample->sequenceMember' equal to a null
pointer is always false [-Wtautological-pointer-compare]
```

The generated code for that method has been fixed and the warning no longer appears when compiling the method.

[RTI Issue ID CODEGENII-438]

3.41 Invalid Java Code Generated when Optional Member Immediately Followed Key Member

The Java code generated for types in which the last key member was followed by an optional member was wrong and did not compile. For example:

```
struct Message {
    long messageId; //@key
    string<255> assetManagerId; //@Optional
};
```

This problem has been resolved.

[RTI Issue ID CODEGENII-447]

4 Known Issues

4.1 Request and Reply Topics Must be Created with Types Generated by Code Generator—C API Only

When using the C API to create Request and Reply Topics, these topics must use data types that have been generated by *Code Generator.* Other APIs support using built-in types and Dynamic-Data types.

[RTI Issue ID BIGPINE-537]

4.2 Code Generation for Inline Nested Structures, Unions, and Valuetypes not Supported

Code generation for inline nested structures, unions, and valuetypes is not supported. For example, *Code Generator* will produce erroneous code for these structures:

IDL:

```
struct Outer {
    short outer_short;
    struct Inner {
        char inner_char;
        short inner_short;
    } outer_nested_inner;
};
```

XML:

```
<struct name="Outer">
    <member name="outer_short" type="short"/>
    <struct name="Inner">
        <member name="inner_char" type="char"/>
        <member name="inner_short" type="short"/>
    </struct>
</struct>
```

[RTI Issue ID CODEGEN-54]

4.3 Classes and Types Defined in Some .NET Namespaces cannot be used to Define User Data Types

The name of the classes and types defined in the following .NET namespaces cannot be used to define user data types:

- System
- System::Collections
- DDS

For example, if you try to define the following enumeration in IDL:

```
enum StatusKind{
    TSK_Unknown,
    TSK_Auto
};
```

The compilation of the generated CPP/CLI code will fail with the following error message:

```
error C2872: 'StatusKind' : ambiguous symbol
```

The reason for this error message is that the enumeration StatusKind is also defined in the DDS namespace and the generated code includes this namespace using the "using" directive:

```
using namespace DDS;
```

The rational behind using the "using" directive was to make the generated code shorter and more readable.

[RTI Issue ID CODEGEN-547]

4.4 To Declare Arrays as Optional in C/C++, They Must be Aliased

When generating C or C++ code, arrays cannot be declared as optional unless they are aliased.

[RTI Issue ID CODEGEN-604]

4.5 Unable to Detect if Optional Member is Inside Aggregated Key Member

Code Generator cannot detect if an optional member is inside an aggregated key member.

[RTI Issue ID CODEGEN-605]

4.6 .NET Code Generation for Multi-dimensional Arrays of Sequences not Supported

The .NET code generated by *Code Generator* for multi-dimensional arrays of sequences is not correct and will not compile.

For example:

```
struct MyStruct {
    sequence<short, 4> m1[3][2];
};
```

[RTI Issue IDs CODEGENII-317, CODEGEN-376]

5 Third-Party Licenses

Portions of *RTI Code Generator* were developed using:

- Apache log4j™ from the Apache Software Foundation (http://logging.apache.org/log4j/)
- Apache Velocity™ from the Apache Software Foundation (http://velocity.apache.org/)
- ANTLR v3 (http://www.antlr3.org/)

5.1 Apache Software License Version 2.0

Apache License

Version 2.0, January 2004

http://www.apache.org/licenses/

TERMS AND CONDITIONS FOR USE, REPRODUCTION, AND DISTRIBUTION

1. Definitions.

"License" shall mean the terms and conditions for use, reproduction, and distribution as defined by Sections 1 through 9 of this document. "Licensor" shall mean the copyright owner or entity authorized by the copyright owner that is granting the License.

"Legal Entity" shall mean the union of the acting entity and all other entities that control, are controlled by, or are under common control with that entity. For the purposes of this definition, "control" means (i) the power, direct or indirect, to cause the direction or management of such entity, whether by contract or otherwise, or (ii) ownership of fifty percent (50%) or more of the outstanding shares, or (iii) beneficial ownership of such entity.

"You" (or "Your") shall mean an individual or Legal Entity exercising permissions granted by this License.

"Source" form shall mean the preferred form for making modifications, including but not limited to software source code, documentation source, and configuration files.

"Object" form shall mean any form resulting from mechanical transformation or translation of a Source form, including but not limited to compiled object code, generated documentation, and conversions to other media types.

"Work" shall mean the work of authorship, whether in Source or Object form, made available under the License, as indicated by a copyright notice that is included in or attached to the work (an example is provided in the Appendix below).

"Derivative Works" shall mean any work, whether in Source or Object form, that is based on (or derived from) the Work and for which the editorial revisions, annotations, elaborations, or other modifications represent, as a whole, an original work of authorship. For the purposes of this License, Derivative Works shall not include works that remain separable from, or merely link (or bind by name) to the interfaces of, the Work and Derivative Works thereof.

"Contribution" shall mean any work of authorship, including the original version of the Work and any modifications or additions to that Work or Derivative Works thereof, that is intentionally submitted to Licensor for inclusion in the Work by the copyright owner or by an individual or Legal Entity authorized to submit on behalf of the copyright owner. For the purposes of this definition, "submitted" means any form of electronic, verbal, or written communication sent to the Licensor or its representatives, including but not limited to communication on electronic mailing lists, source code control systems, and issue tracking systems that are managed by, or on behalf of, the Licensor for the purpose of discussing and improving the Work, but excluding communication that is conspicuously marked or otherwise designated in writing by the copyright owner as "Not a Contribution."

"Contributor" shall mean Licensor and any individual or Legal Entity on behalf of whom a Contribution has been received by Licensor and subsequently incorporated within the Work.

2. Grant of Copyright License. Subject to the terms and conditions of this License, each Contributor hereby grants to You a perpetual, worldwide, non-exclusive, no-charge, royalty-free, irrevocable copyright license to reproduce, prepare Derivative Works of, publicly display, publicly perform, sublicense, and distribute the Work and such Derivative Works in Source or Object form.

3. Grant of Patent License. Subject to the terms and conditions of this License, each Contributor hereby grants to You a perpetual, worldwide, non-exclusive, no-charge, royalty-free, irrevocable (except as stated in this section) patent license to make, have made, use, offer to sell, sell, import, and otherwise transfer the Work, where such license applies only to those patent claims licensable by such Contributor that are necessarily infringed by their Contribution(s) alone or by combination of their Contribution(s) with the Work to which such Contribution(s) was submitted. If You institute patent litigation against any entity (including a cross-claim or counterclaim in a lawsuit) alleging that the Work or a Contribution incorporated within the Work constitutes direct or contributory patent infringement, then any patent licenses granted to You under this License for that Work shall terminate as of the date such litigation is filed.

4. Redistribution. You may reproduce and distribute copies of the Work or Derivative Works thereof in any medium, with or without modifications, and in Source or Object form, provided that You meet the following conditions:

(a) You must give any other recipients of the Work or Derivative Works a copy of this License; and

(b) You must cause any modified files to carry prominent notices stating that You changed the files; and

(c) You must retain, in the Source form of any Derivative Works that You distribute, all copyright, patent, trademark, and attribution notices from the Source form of the Work, excluding those notices that do not pertain to any part of the Derivative Works; and

(d) If the Work includes a "NOTICE" text file as part of its distribution, then any Derivative Works that You distribute must include a readable copy of the attribution notices contained within such NOTICE file, excluding those notices that do not pertain to any part of the Derivative Works, in at least one of the following places: within a NOTICE text file distributed as part of the Derivative Works; within the Source form or documentation, if provided along with the Derivative Works; or, within a display generated by the Derivative Works, if and wherever such third-party notices normally appear. The contents of the NOTICE file are for informational purposes only and do not modify the License. You may add Your own attribution notices within Derivative Works that You distribute, alongside or as an addendum to the NOTICE text from the Work, provided that such additional attribution notices cannot be construed as modifying the License.

You may add Your own copyright statement to Your modifications and may provide additional or different license terms and conditions for use, reproduction, or distribution of Your modifications, or for any such Derivative Works as a whole, provided Your use, reproduction, and distribution of the Work otherwise complies with the conditions stated in this License.

5. Submission of Contributions. Unless You explicitly state otherwise, any Contribution intentionally submitted for inclusion in the Work by You to the Licensor shall be under the terms and conditions of this License, without any additional terms or conditions. Notwithstanding the above, nothing herein shall supersede or modify the terms of any separate license agreement you may have executed with Licensor regarding such Contributions.

6. Trademarks. This License does not grant permission to use the trade names, trademarks, service marks, or product names of the Licensor, except as required for reasonable and customary use in describing the origin of the Work and reproducing the content of the NOTICE file.

7. Disclaimer of Warranty. Unless required by applicable law or agreed to in writing, Licensor provides the Work (and each Contributor provides its Contributions) on an "AS IS" BASIS, WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied, including, without limitation, any warranties or conditions of TITLE, NON-INFRINGEMENT, MERCHANTABILITY, or FITNESS FOR A PARTICULAR PURPOSE. You are solely responsible for

determining the appropriateness of using or redistributing the Work and assume any risks associated with Your exercise of permissions under this License.

8. Limitation of Liability. In no event and under no legal theory, whether in tort (including negligence), contract, or otherwise, unless required by applicable law (such as deliberate and grossly negligent acts) or agreed to in writing, shall any Contributor be liable to You for damages, including any direct, indirect, special, incidental, or consequential damages of any character arising as a result of this License or out of the use or inability to use the Work (including but not limited to damages for loss of goodwill, work stoppage, computer failure or malfunction, or any and all other commercial damages or losses), even if such Contributor has been advised of the possibility of such damages.

9. Accepting Warranty or Additional Liability. While redistributing the Work or Derivative Works thereof, You may choose to offer, and charge a fee for, acceptance of support, warranty, indemnity, or other liability obligations and/or rights consistent with this License. However, in accepting such obligations, You may act only on Your own behalf and on Your sole responsibility, not on behalf of any other Contributor, and only if You agree to indemnify, defend, and hold each Contributor harmless for any liability incurred by, or claims asserted against, such Contributor by reason of your accepting any such warranty or additional liability.

END OF TERMS AND CONDITIONS

5.2 ANTLR 3 License

[The BSD License]

Copyright (c) 2010 Terence Parr

All rights reserved.

Redistribution and use in source and binary forms, with or without modification, are permitted provided that the following conditions are met:

- Redistributions of source code must retain the above copyright notice, this list of conditions and the following disclaimer.
- Redistributions in binary form must reproduce the above copyright notice, this list of conditions and the following disclaimer in the documentation and/or other materials provided with the distribution.
- Neither the name of the author nor the names of its contributors may be used to endorse or promote products derived from this software without specific prior written permission.

THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT OWNER OR CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.