RTI Persistence Service

Version 7.7.0

Chapter 1

RTI Persistence Service for RTI Connext Messaging

Real-Time Innovations, Inc.

1.1 Available Documentation

This document contains:

- **Overview** (p. 2)
- **Using RTI Persistence Service** (p. 2)
- **Running RTI Persistence Service** (p. 2)
- **Using the RTI Persistence Service as a Library** (p. 3)

For additional information, please see the following PDF documents:

- **`RTI Persistence Service Release Notes`**
- **`Core Libraries User's Manual`** (RTI Persistence Service section)

Please do not hesitate to contact RTI with questions or comments about this release. We welcome any input on how to improve RTI Persistence Service to suit your needs.

1.2 Overview

RTI Persistence Service is an optional package that saves data samples so that they can be delivered to subscribing applications that join the system at a later time – even if the publishing application has already terminated.

RTI Persistence Service runs as a separate application; you can run it on the same node as the publishing application, the subscribing application, or some other node in the network.

When configured to run in PERSISTENT mode, RTI Persistence Service uses the filesystem to persist data samples. For each persistent topic, it collects all the data written by the corresponding persistent DataWriters and stores them into persistent storage.

When configured to run in TRANSIENT mode, RTI Persistence Service stores the data in memory.

1.3 Using RTI Persistence Service

To use RTI Persistence Service in your RTI Connext Messaging application, you need to modify the **DURABILITY QosPolicy** of the DataReader and/or DataWriter. You may optionally modify the **DURABILITY_SERVICE QosPolicy**, which can be used to configure RTI Persistence Service.

For more information, please see `RTI Persistence Service`, in the Core Libraries User's Manual.

1.4 Running RTI Persistence Service

You can run RTI Persistence Service on any node in the network; it does not have to be run on the same node as the publishing or subscribing applications for which it is saving/delivering data. If you run RTI Persistence Service on a separate node, however, make sure that the other applications can find it during the discovery process – that is, it must be in one of the **NDDS_DISCOVERY_PEERS** lists.

A script to run the RTI Persistence Service executable is located in **$NDDSHOME/bin**.

For details, see `Running RTI Persistence Service`, in the Core Libraries User's Manual.

1.5 Finding Example Applications

A simple example is available to show the how data can be persisted with RTI Persistence Service.

C: See the example in <path to examples>/persistence_service/c/hello_world_persistence.

C++: See the example in <path to examples>/persistence_service/c++/hello_world_persistence.

Java: See the example in <path to examples>/persistence_service/java/hello_world_persistence.

"path to examples" is your rti_workspace directory, which is in your Windows documents folder, Linux home/<your user name> directory, or macOS Users/<your user name> directory.

1.6 Configuring RTI Persistence Service

See `Configuring Persistence Service`, in the Core Libraries User's Manual, for complete information.

1.7 Using the RTI Persistence Service as a Library

Persistence can be deployed as a C library linked into your application in selected architectures.

Refer to the `Platform Notes` for the list of supported architectures.

You can configure, start, and stop the service using the API described here:

- **Library API** (p. 11)

Chapter 2

Multi-threading safety

Global RTI_PersistenceService_get_participant_qos (p. 14) (const struct RTI_PersistenceService (p. 93) *self, struct DDS_DomainParticipantQos *domain_participant_qos, const char *participant_name)

This method is thread safe as long as the **RTI_PersistenceService** (p. 93) instance is not deleted at the same time by calling **RTI_PersistenceService_delete** (p. 19). Otherwise it is thread safe across calls to any other QoS related Library API methods.

Global RTI_PersistenceService_get_publisher_qos (p. 15) (const struct RTI_PersistenceService (p. 93) *self, struct DDS_PublisherQos *publisher_qos, const char *participant_name, const char *persistence_group_name)

This method is thread safe as long as the **RTI_PersistenceService** (p. 93) instance is not deleted at the same time by calling **RTI_PersistenceService_delete** (p. 19). Otherwise it is thread safe across calls to any other QoS related Library API methods.

Global RTI_PersistenceService_get_subscriber_qos (p. 17) (const struct RTI_PersistenceService (p. 93) *self, struct DDS_SubscriberQos *subscriber_qos, const char *participant_name, const char *persistence_group_name)

This method is thread safe as long as the **RTI_PersistenceService** (p. 93) instance is not deleted at the same time by calling **RTI_PersistenceService_delete** (p. 19). Otherwise it is thread safe across calls to any other QoS related Library API methods.

Global RTI_PersistenceService_set_participant_qos (p. 15) (struct RTI_PersistenceService (p. 93) *self, const char *participant_name, const struct DDS_DomainParticipantQos *domain_participant_qos)

This method is thread safe as long as the **RTI_PersistenceService** (p. 93) instance is not deleted at the same time by calling **RTI_PersistenceService_delete** (p. 19). Otherwise it is thread safe across calls to any other QoS related Library API methods.

Global RTI_PersistenceService_set_publisher_qos (p. 16) (struct RTI_PersistenceService (p. 93) *self, const char *participant_name, const char *persistence_group_name, const struct DDS_PublisherQos *publisher_qos)

This method is thread safe as long as the **RTI_PersistenceService** (p. 93) instance is not deleted at the same time by calling **RTI_PersistenceService_delete** (p. 19). Otherwise it is thread safe across calls to any other QoS related Library API methods.

Global RTI_PersistenceService_set_subscriber_qos (p. 17) (struct RTI_PersistenceService (p. 93) *self, const char *participant_name, const char *persistence_group_name, const struct DDS_SubscriberQos *subscriber_qos)

This method is thread safe as long as the **RTI_PersistenceService** (p. 93) instance is not deleted at the same time by calling **RTI_PersistenceService_delete** (p. 19). Otherwise it is thread safe across calls to any other QoS related Library API methods.

Global RTI_PersistenceService_start (p. 19) (struct RTI_PersistenceService (p. 93) *self)

This method is not thread-safe. Calling this method from different threads for the same Persistence Service instance may result in undefined behavior.

Global RTI_PersistenceService_stop (p. 20) (struct RTI_PersistenceService (p. 93) *self)

This method is not thread-safe. Calling this method from different threads for the same Persistence Service instance may result in undefined behavior.

Chapter 3

Module Index

3.1 Modules

Here is a list of all modules:

Chapter 4

Data Structure Index

4.1 Data Structures

Here are the data structures with brief descriptions:

Chapter 5

Module Documentation

5.1 Persistence Service Examples

Persistence Service Examples.

Modules

- **C Example**

  Persistence Example Using C.
- **C++ Example**

  Persistence Example Using C++.
- **Java Example**

  Persistence Example Using Java.

5.1.1 Detailed Description

Persistence Service Examples.

5.2 Library API

This API allows you to embed RTI Persistence Service in your application.

Modules

- **Version API**

Data Structures

- struct **RTI_PersistenceService**

  RTI Persistence Service.
- struct **RTI_PersistenceServiceProperty**

  Configuration of RTI Persistence Service.

Functions

- DDS_Boolean **RTI_PersistenceService_get_participant_qos** (const struct **RTI_PersistenceService** *self, struct DDS_DomainParticipantQos *domain_participant_qos, const char *participant_name)

  Get the DDS_DomainParticipantQos for the internal DDS_DomainParticipant created by RTI Persistence Service.
- DDS_Boolean **RTI_PersistenceService_set_participant_qos** (struct **RTI_PersistenceService** *self, const char *participant_name, const struct DDS_DomainParticipantQos *domain_participant_qos)

  Set the DDS_DomainParticipantQos for the internal DDS_DomainParticipant created by RTI Persistence Service.
- DDS_Boolean **RTI_PersistenceService_get_publisher_qos** (const struct **RTI_PersistenceService** *self, struct DDS_PublisherQos *publisher_qos, const char *participant_name, const char *persistence_group_name)

  Get the DDS_PublisherQos for the internal DDS_Publisher created by RTI Persistence Service.
- DDS_Boolean **RTI_PersistenceService_set_publisher_qos** (struct **RTI_PersistenceService** *self, const char *participant_name, const char *persistence_group_name, const struct DDS_PublisherQos *publisher_qos)

  Set the DDS_PublisherQos for the internal DDS_Publisher created by RTI Persistence Service.
- DDS_Boolean **RTI_PersistenceService_get_subscriber_qos** (const struct **RTI_PersistenceService** *self, struct DDS_SubscriberQos *subscriber_qos, const char *participant_name, const char *persistence_group↩ _name)

  Get the DDS_SubscriberQos for the internal DDS_Subscriber created by RTI Persistence Service.
- DDS_Boolean **RTI_PersistenceService_set_subscriber_qos** (struct **RTI_PersistenceService** *self, const char *participant_name, const char *persistence_group_name, const struct DDS_SubscriberQos *subscriber↩ _qos)

  Set the DDS_SubscriberQos for the internal DDS_Subscriber created by RTI Persistence Service.
- DDS_Boolean **RTI_PersistenceService_is_started** (struct **RTI_PersistenceService** *self)

  *Indicates if RTI Persistence Service was started by calling **RTI_PersistenceService_start** (p. 19).*
- struct **RTI_PersistenceService** * **RTI_PersistenceService_new** (struct **RTI_PersistenceServiceProperty** *property)

  Create a new RTI Persistence Service instance.
- void **RTI_PersistenceService_delete** (struct **RTI_PersistenceService** *self)

  Stop and delete RTI Persistence Service instance.
- DDS_Boolean **RTI_PersistenceService_start** (struct **RTI_PersistenceService** *self)

  Start RTI Persistence Service.
- DDS_Boolean **RTI_PersistenceService_stop** (struct **RTI_PersistenceService** *self)

  Stop RTI Persistence Service.
- DDS_Boolean **RTI_PersistenceService_initialize_globals** (void)

  Initialize the RTI Persistence Service globals.
- void **RTI_PersistenceService_finalize_globals** (void)

  Finalize the RTI Persistence Service globals.

Variables

- const struct **RTI_PersistenceServiceProperty** **RTI_PersistenceServiceProperty_INITIALIZER**

  *The initial values for an **RTI_PersistenceServiceProperty** (p. 93) instance.*
- const int **RTI_PERSISTENCE_SERVICE_LOG_VERBOSITY_INFO**

  Verbosity level: exceptions + warnings + info.
- const int **RTI_PERSISTENCE_SERVICE_LOG_VERBOSITY_WARNINGS**

  Verbosity level: exceptions + warnings.
- const int **RTI_PERSISTENCE_SERVICE_LOG_VERBOSITY_EXCEPTIONS**

  Verbosity level: exceptions.
- const int **RTI_PERSISTENCE_SERVICE_LOG_VERBOSITY_SILENT**

  Verbosity level: silent.
- const int **RTI_PERSISTENCE_SERVICE_LOG_VERBOSITY_IGNORE**

  Verbosity level: ignore.

5.2.1 Detailed Description

This API allows you to embed RTI Persistence Service in your application.

The API allows you to create, configure and start RTI Persistence Service instances from your application. The following code shows the typical use of the API:

```
struct RTI_PersistenceServiceProperty property =
        RTI_PersistenceServiceProperty_INITIALIZER;
struct RTI_PersistenceService * service = NULL;
property.cfg_file = "my_persistence_service_cfg.xml";
property.cfg_name = "default";
...
if (!RTI_PersistenceService_initialize_globals()) {
   printf("Error ... \n");
   return -1;
}
service = RTI_PersistenceService_new(&property);
if (service == NULL) {
   printf("Error ... \n");
   RTI_PersistenceService_finalize_globals();
   return -1;
}
if (!RTI_PersistenceService_start(service)) {
   printf("Error ... \n");
   RTI_PersistenceService_delete(service);
   RTI_PersistenceService_finalize_globals();
   return -1;
}
while(keep_running) {
    sleep();
    ...
}
RTI_PersistenceService_delete(service);
RTI_PersistenceService_finalize_globals();
return 0;
```

Instead of a file, you can use XML strings to configure RTI Persistence Service. See **RTI_PersistenceServiceProperty** (p. 93) for more information.

To build your application you need to link with the RTI Persistence Service library in **$NDDSHOME/lib/**<**architecture**>/

An example is provided at **https://github.com/rticommunity/rticonnextdds-examples/tree/master/exampl**
_service/library_api.

	Linux Systems	Windows Systems
Link Libraries (Compile Time)	librtipersistenceservice.so	rtipersistenceservice.dll
Shared Libraries (Run Time)	librtidlc.so	rtidlc.dll
	libnddsc.so	nddsc.dll
	libnddscore.so	nddscore.dll
	librtisqlite.so	librtisqlite.dll
	librtimonitoring.so	librtimonitoring.dll
Static Libraries	librtipersistenceservicez.a	rtipersistenceservicez.lib
	librtidlc.a	rtidlc.lib
	libnddsc.a	nddsc.lib
	libnddscore.a	nddscore.lib
	librtisqlite.a	librtisqlite.lib
Headers	persistence/persistence_↩ service.h	

5.2.1.0.1 Development Requirements **NOTE:** If you are using debug libraries, remember to add the 'd' suffix to the library name. For shared libraries on macOS systems, the library name is the same as that for Linux except the extension is '.dylib' instead of '.so'.

5.2.2 Function Documentation

5.2.2.1 RTI_PersistenceService_get_participant_qos()

```
DDS_Boolean RTI_PersistenceService_get_participant_qos (
            const struct  RTI_PersistenceService * self,
            struct DDS_DomainParticipantQos * domain_participant_qos,
            const char * participant_name )
```

Get the DDS_DomainParticipantQos for the internal DDS_DomainParticipant created by RTI Persistence Service.

Parameters

in	*self*	**RTI_PersistenceService** (p. 93) instance created with **RTI_PersistenceService_new** (p. 18)
out	*domain_participant_qos*	DDS_DomainParticipantQos the value is copied into. This method returns the value of the first DDS_DomainParticipant found under the <participant> tag identified by participant_name.
in	*participant_name*	Identifies a <participant> tag from the XML configuration

Multi-threading safety This method is thread safe as long as the **RTI_PersistenceService** (p. 93) instance is not deleted at the same time by calling **RTI_PersistenceService_delete** (p. 19). Otherwise it is thread safe across calls to any other QoS related Library API methods.

Returns

DDS_BOOLEAN_TRUE on success; DDS_BOOLEAN_FALSE otherwise.

5.2.2.2 RTI_PersistenceService_set_participant_qos()

```
DDS_Boolean RTI_PersistenceService_set_participant_qos (
            struct  RTI_PersistenceService * self,
            const char * participant_name,
            const struct DDS_DomainParticipantQos * domain_participant_qos )
```

Set the DDS_DomainParticipantQos for the internal DDS_DomainParticipant created by RTI Persistence Service.

Parameters

in	*self*	**RTI_PersistenceService** (p. 93) instance created with **RTI_PersistenceService_new** (p. 18)
in	*participant_name*	Identifies a <participant> tag from the XML configuration
in	*domain_participant_qos*	DDS_DomainParticipantQos value to set for all DDS_DomainParticipant under the <participant> tag identified by participant_name and participant_kind

Multi-threading safety This method is thread safe as long as the **RTI_PersistenceService** (p. 93) instance is not deleted at the same time by calling **RTI_PersistenceService_delete** (p. 19). Otherwise it is thread safe across calls to any other QoS related Library API methods.

Returns

DDS_BOOLEAN_TRUE on success; DDS_BOOLEAN_FALSE otherwise.

5.2.2.3 RTI_PersistenceService_get_publisher_qos()

```
DDS_Boolean RTI_PersistenceService_get_publisher_qos (
            const struct  RTI_PersistenceService * self,
            struct DDS_PublisherQos * publisher_qos,
            const char * participant_name,
            const char * persistence_group_name )
```

Get the DDS_PublisherQos for the internal DDS_Publisher created by RTI Persistence Service.

Parameters

in	*self*	**RTI_PersistenceService** (p. 93) instance created with **RTI_PersistenceService_new** (p. 18)
out	*publisher_qos*	DDS_PublisherQos the value is copied into. This method returns the value of the first DDS_Publisher found under the <participant>/<persistence_group> tag identified by participant_name and persistence_group_name.
in	*participant_name*	Identifies a <participant> tag from the XML configuration
in	*persistence_group_name*	Identifies a <persistence_group> tag from the XML configuration

Multi-threading safety This method is thread safe as long as the **RTI_PersistenceService** (p. 93) instance is not deleted at the same time by calling **RTI_PersistenceService_delete** (p. 19). Otherwise it is thread safe across calls to any other QoS related Library API methods.

Returns

DDS_BOOLEAN_TRUE on success; DDS_BOOLEAN_FALSE otherwise.

5.2.2.4 RTI_PersistenceService_set_publisher_qos()

```
DDS_Boolean RTI_PersistenceService_set_publisher_qos (
            struct  RTI_PersistenceService * self,
            const char * participant_name,
            const char * persistence_group_name,
            const struct DDS_PublisherQos * publisher_qos )
```

Set the DDS_PublisherQos for the internal DDS_Publisher created by RTI Persistence Service.

Parameters

in	*self*	**RTI_PersistenceService** (p. 93) instance created with **RTI_PersistenceService_new** (p. 18)
in	*participant_name*	Identifies a <participant> tag from the XML configuration
in	*persistence_group_name*	Identifies a <persistence_group> tag from the XML configuration
in	*publisher_qos*	DDS_PublisherQos value to set for all DDS_Publisher under the <participant>/<persistence_group> tag identified by participant_name and persistence_group_name. This value is applied to all the DDS_Publisher created per DDS_Topic if <single_publisher> is set to false.

Multi-threading safety This method is thread safe as long as the **RTI_PersistenceService** (p. 93) instance is not deleted at the same time by calling **RTI_PersistenceService_delete** (p. 19). Otherwise it is thread safe across calls to any other QoS related Library API methods.

Returns

DDS_BOOLEAN_TRUE on success; DDS_BOOLEAN_FALSE otherwise.

5.2.2.5 RTI_PersistenceService_get_subscriber_qos()

```
DDS_Boolean RTI_PersistenceService_get_subscriber_qos (
            const struct  RTI_PersistenceService * self,
            struct DDS_SubscriberQos * subscriber_qos,
            const char * participant_name,
            const char * persistence_group_name )
```

Get the DDS_SubscriberQos for the internal DDS_Subscriber created by RTI Persistence Service.

Parameters

in	*self*	**RTI_PersistenceService** (p. 93) instance created with **RTI_PersistenceService_new** (p. 18)
out	*subscriber_qos*	DDS_SubscriberQos the value is copied into. This method returns the value of the first DDS_Subscriber found under the <participant>/<persistence_group> tag identified by participant_name and persistence_group_name.
in	*participant_name*	Identifies a <participant> tag from the XML configuration
in	*persistence_group_name*	Identifies a <persistence_group> tag from the XML configuration

Multi-threading safety This method is thread safe as long as the **RTI_PersistenceService** (p. 93) instance is not deleted at the same time by calling **RTI_PersistenceService_delete** (p. 19). Otherwise it is thread safe across calls to any other QoS related Library API methods.

Returns

DDS_BOOLEAN_TRUE on success; DDS_BOOLEAN_FALSE otherwise.

5.2.2.6 RTI_PersistenceService_set_subscriber_qos()

```
DDS_Boolean RTI_PersistenceService_set_subscriber_qos (
            struct  RTI_PersistenceService * self,
            const char * participant_name,
            const char * persistence_group_name,
            const struct DDS_SubscriberQos * subscriber_qos )
```

Set the DDS_SubscriberQos for the internal DDS_Subscriber created by RTI Persistence Service.

Parameters

in	*self*	**RTI_PersistenceService** (p. 93) instance created with **RTI_PersistenceService_new** (p. 18)
in	*participant_name*	Identifies a <participant> tag from the XML configuration
in	*persistence_group_name*	Identifies a <persistence_group> tag from the XML configuration
in	*subscriber_qos*	DDS_SubscriberQos value to set for all DDS_Subscriber under the <participant>/<persistence_group> tag identified by participant_name and persistence_group_name. This value is applied to all the DDS_Subscriber created per DDS_Topic if <single_subscriber> is set to false.

Multi-threading safety This method is thread safe as long as the **RTI_PersistenceService** (p. 93) instance is not deleted at the same time by calling **RTI_PersistenceService_delete** (p. 19). Otherwise it is thread safe across calls to any other QoS related Library API methods.

Returns

DDS_BOOLEAN_TRUE on success; DDS_BOOLEAN_FALSE otherwise.

5.2.2.7 RTI_PersistenceService_is_started()

```
DDS_Boolean RTI_PersistenceService_is_started (
            struct  RTI_PersistenceService * self )
```

Indicates if RTI Persistence Service was started by calling **RTI_PersistenceService_start** (p. 19).

Parameters

in	*self*	**RTI_PersistenceService** (p. 93) instance created with **RTI_PersistenceService_new** (p. 18)

Returns

DDS_BOOLEAN_TRUE if **RTI_PersistenceService** (p. 93) was started; DDS_BOOLEAN_FALSE otherwise.

5.2.2.8 RTI_PersistenceService_new()

```
struct  RTI_PersistenceService * RTI_PersistenceService_new (
            struct  RTI_PersistenceServiceProperty * property )
```

Create a new RTI Persistence Service instance.

Parameters

in	*property*	The properties to configure RTI Persistence Service

This parameter is copied internally, so the user is responsible for releasing any memory allocated inside this structure.

Returns

An instance of **RTI_PersistenceService** (p. 93) on success; NULL otherwise.

5.2.2.9 RTI_PersistenceService_delete()

```
void RTI_PersistenceService_delete (
            struct  RTI_PersistenceService * self )
```

Stop and delete RTI Persistence Service instance.

See also

RTI_PersistenceService_stop (p. 20)

Parameters

in	*self*	**RTI_PersistenceService** (p. 93) instance created with **RTI_PersistenceService_new** (p. 18)

5.2.2.10 RTI_PersistenceService_start()

```
DDS_Boolean RTI_PersistenceService_start (
            struct  RTI_PersistenceService * self )
```

Start RTI Persistence Service.

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

Parameters

in	*self*	**RTI_PersistenceService** (p. 93) instance created with **RTI_PersistenceService_new** (p. 18)

Returns

DDS_BOOLEAN_TRUE on success; DDS_BOOLEAN_FALSE otherwise.

Multi-threading safety This method is not thread-safe. Calling this method from different threads for the same Persistence Service instance may result in undefined behavior.

5.2.2.11 RTI_PersistenceService_stop()

```
DDS_Boolean RTI_PersistenceService_stop (
            struct  RTI_PersistenceService * self )
```

Stop RTI Persistence Service.

This function will not return execution control until the instance is fully stopped.

Parameters

in	*self*	**RTI_PersistenceService** (p. 93) instance created with **RTI_PersistenceService_new** (p. 18)

Returns

DDS_BOOLEAN_TRUE on success; DDS_BOOLEAN_FALSE otherwise.

Multi-threading safety This method is not thread-safe. Calling this method from different threads for the same Persistence Service instance may result in undefined behavior.

5.2.2.12 RTI_PersistenceService_initialize_globals()

```
DDS_Boolean RTI_PersistenceService_initialize_globals (
            void  )
```

Initialize the RTI Persistence Service globals.

This function will initialize RTI Persistence Service globals. This function must be called before creating any RTI Persistence Service instance.

Returns

DDS_BOOLEAN_TRUE on success; DDS_BOOLEAN_FALSE otherwise.

5.2.2.13 RTI_PersistenceService_finalize_globals()

```
void RTI_PersistenceService_finalize_globals (
            void  )
```

Finalize the RTI Persistence Service globals.

This function will finalize RTI Persistence Service globals. This function must be called after all the instances of RTI Persistence Service have been deleted.

5.2.3 Variable Documentation

5.2.3.1 RTI_PersistenceServiceProperty_INITIALIZER

```
const struct  RTI_PersistenceServiceProperty RTI_PersistenceServiceProperty_INITIALIZER
```

The initial values for an **RTI_PersistenceServiceProperty** (p. 93) instance.

5.2.3.2 RTI_PERSISTENCE_SERVICE_LOG_VERBOSITY_INFO

```
const int RTI_PERSISTENCE_SERVICE_LOG_VERBOSITY_INFO
```

Verbosity level: exceptions + warnings + info.

5.2.3.3 RTI_PERSISTENCE_SERVICE_LOG_VERBOSITY_WARNINGS

```
const int RTI_PERSISTENCE_SERVICE_LOG_VERBOSITY_WARNINGS
```

Verbosity level: exceptions + warnings.

5.2.3.4 RTI_PERSISTENCE_SERVICE_LOG_VERBOSITY_EXCEPTIONS

```
const int RTI_PERSISTENCE_SERVICE_LOG_VERBOSITY_EXCEPTIONS
```

Verbosity level: exceptions.

5.2.3.5 RTI_PERSISTENCE_SERVICE_LOG_VERBOSITY_SILENT

```
const int RTI_PERSISTENCE_SERVICE_LOG_VERBOSITY_SILENT
```

Verbosity level: silent.

5.2.3.6 RTI_PERSISTENCE_SERVICE_LOG_VERBOSITY_IGNORE

```
const int RTI_PERSISTENCE_SERVICE_LOG_VERBOSITY_IGNORE
```

Verbosity level: ignore.

This value is meant to be used only with the dds_verbosity field. It instructs RTI Persistence Service to use the existing value set for DDS_DomainParticipantFactory logging verbosity.

5.3 Version API

Typedefs

- typedef struct DDS_ProductVersion_t **RTI_PersistenceServiceLibraryVersion**
  Persistence Library Version.

Functions

- const **RTI_PersistenceServiceLibraryVersion** * **RTI_PersistenceService_get_library_version** (void)
  Get the library version.
- const char * **RTI_PersistenceService_get_build_version_string** (void)
  Get the build version.

5.3.1 Detailed Description

5.3.2 Typedef Documentation

5.3.2.1 RTI_PersistenceServiceLibraryVersion

```
typedef struct DDS_ProductVersion_t  RTI_PersistenceServiceLibraryVersion
```

Persistence Library Version.

5.3.3 Function Documentation

5.3.3.1 RTI_PersistenceService_get_library_version()

```
const  RTI_PersistenceServiceLibraryVersion * RTI_PersistenceService_get_library_version (
            void  )
```

Get the library version.

5.3.3.2 RTI_PersistenceService_get_build_version_string()

```
const char * RTI_PersistenceService_get_build_version_string (
            void  )
```

Get the build version.

5.4 C Example

Persistence Example Using C.

Modules

- **HelloWorld.idl**
- **HelloWorld.c**
- **HelloWorld_publisher.c**
- **HelloWorld_subscriber.c**
- **HelloWorldPlugin.c**
- **HelloWorldSupport.c**

5.4.1 Detailed Description

Persistence Example Using C.

See README.txt for details.

[$(NDDSHOME)/example/C/helloWorldPersistence/README.txt]

5.5 HelloWorld.idl

[$(NDDSHOME)/example/C/helloWorldPersistence/HelloWorld.idl]

```
struct HelloWorld {
    long data;
};
```

5.6 HelloWorld.c

[$(NDDSHOME)/example/C/helloWorldPersistence/HelloWorld.h]

```
/*
WARNING: THIS FILE IS AUTO-GENERATED. DO NOT MODIFY.

This file was generated from HelloWorld.idl
using RTI Code Generator (rtiddsgen) version 4.7.0.
The rtiddsgen tool is part of the RTI Connext DDS distribution.
For more information, type 'rtiddsgen -help' at a command shell
or consult the Code Generator User's Manual.
*/
#ifndef HelloWorld_1436885481_h
#define HelloWorld_1436885481_h
#ifndef NDDS_STANDALONE_TYPE
#ifndef ndds_c_h
#include "ndds/ndds_c.h"
#endif
#include "cdr/cdr_typeCode.h"
#else
#include "ndds_standalone_type.h"
#endif
#ifdef __cplusplus
extern "C" {
    #endif
    extern const char *HelloWorldTYPENAME;
    typedef struct HelloWorld
    {
        DDS_Long data;
    } HelloWorld ;
    #if defined(NDDS_USER_DLL_EXPORT) && defined(RTI_WIN32)
    #undef NDDSUSERDllExport
    #define NDDSUSERDllExport __declspec(dllexport)
    #endif
    #if !defined(RTI_WIN32) && defined(NDDS_USER_SYMBOL_EXPORT)
    #undef NDDSUSERDllExport
    #define NDDSUSERDllExport __attribute__((visibility("default")))
    #endif
    #ifndef NDDS_STANDALONE_TYPE
    NDDSUSERDllExport DDS_TypeCode * HelloWorld_get_typecode(void); /* Type code */
    NDDSUSERDllExport RTIXCdrTypePlugin *HelloWorld_get_type_plugin_info(void);
    NDDSUSERDllExport RTIXCdrSampleAccessInfo *HelloWorld_get_sample_access_info(void);
    #endif
    DDS_SEQUENCE(HelloWorldSeq, HelloWorld);
    NDDSUSERDllExport
    RTIBool HelloWorld_initialize(
        HelloWorld* self);
    NDDSUSERDllExport
    RTIBool HelloWorld_initialize_ex(
        HelloWorld* self,RTIBool allocatePointers,RTIBool allocateMemory);
    NDDSUSERDllExport
    RTIBool HelloWorld_initialize_w_params(
        HelloWorld* self,
        const struct DDS_TypeAllocationParams_t * allocParams);
    NDDSUSERDllExport
    RTIBool HelloWorld_finalize_w_return(
        HelloWorld* self);
    NDDSUSERDllExport
    void HelloWorld_finalize(
        HelloWorld* self);
    NDDSUSERDllExport
    void HelloWorld_finalize_ex(
        HelloWorld* self,RTIBool deletePointers);
    NDDSUSERDllExport
    void HelloWorld_finalize_w_params(
        HelloWorld* self,
```

```
        const struct DDS_TypeDeallocationParams_t * deallocParams);
    NDDSUSERDllExport
    void HelloWorld_finalize_optional_members(
        HelloWorld* self, RTIBool deletePointers);
    NDDSUSERDllExport
    RTIBool HelloWorld_copy(
        HelloWorld* dst,
        const HelloWorld* src);
    #if defined(NDDS_USER_DLL_EXPORT) || defined(NDDS_USER_SYMBOL_EXPORT)
    #undef NDDSUSERDllExport
    #define NDDSUSERDllExport
    #endif
    #ifdef __cplusplus
}
#endif
#endif /* HelloWorld */
```

[$(NDDSHOME)/example/C/helloWorldPersistence/HelloWorld.c]

```
/*
WARNING: THIS FILE IS AUTO-GENERATED. DO NOT MODIFY.

This file was generated from HelloWorld.idl
using RTI Code Generator (rtiddsgen) version 4.7.0.
The rtiddsgen tool is part of the RTI Connext DDS distribution.
For more information, type 'rtiddsgen -help' at a command shell
or consult the Code Generator User's Manual.
*/
#ifndef NDDS_STANDALONE_TYPE
#ifndef ndds_c_h
#include "ndds/ndds_c.h"
#endif
#ifndef dds_c_log_infrastructure_h
#include "dds_c/dds_c_infrastructure_impl.h"
#endif
#ifndef cdr_type_h
#include "cdr/cdr_type.h"
#endif
#include "osapi/osapi_atomic.h"
#else
#include "ndds_standalone_type.h"
#endif
#include "HelloWorld.h"
#ifndef NDDS_STANDALONE_TYPE
#include "HelloWorldPlugin.h"
#endif
/* ========================================================================= */
const char *HelloWorldTYPENAME = "HelloWorld";
#ifndef NDDS_STANDALONE_TYPE
DDS_TypeCode * HelloWorld_get_typecode(void)
{
    static RTI_ATOMIC(RTIBool) is_initialized;
    static DDS_TypeCode_Member HelloWorld_g_tc_members[1]=
    {
        {
            (char *)"data",/* Member name */
            {
                0,/* Representation ID */
                DDS_BOOLEAN_FALSE,/* Is a pointer? */
                -1, /* Bitfield bits */
                NULL/* Member type code is assigned later */
            },
            0, /* Ignored */
            0, /* Ignored */
            0, /* Ignored */
            NULL, /* Ignored */
            RTI_CDR_REQUIRED_MEMBER, /* Is a key? */
            DDS_PUBLIC_MEMBER,/* Member visibility */
            RTICdrTypeCodeAnnotations_INITIALIZER
        }
    };
    static DDS_TypeCode HelloWorld_g_tc =
    {{
            DDS_TK_STRUCT, /* Kind */
            DDS_BOOLEAN_FALSE, /* Ignored */
            -1, /*Ignored*/
            (char *)"HelloWorld", /* Name */
            NULL, /* Ignored */
            0, /* Ignored */
            0, /* Ignored */
            NULL, /* Ignored */
```

```
            1, /* Number of members */
            HelloWorld_g_tc_members, /* Members */
            DDS_VM_NONE, /* Ignored */
            RTICdrTypeCodeAnnotations_INITIALIZER,
            DDS_BOOLEAN_TRUE, /* _isCopyable */
            NULL, /* _sampleAccessInfo: assigned later */
            NULL /* _typePlugin: assigned later */
        }}; /* Type code for HelloWorld*/
    if (RTIOsapiAtomic_load32(&is_initialized, RTI_OSAPI_ATOMIC_MEMORY_ORDER_ACQUIRE)) {
        return &HelloWorld_g_tc;
    }
    HelloWorld_g_tc._data._annotations._allowedDataRepresentationMask = 5;
    HelloWorld_g_tc_members[0]._representation._typeCode =  (RTICdrTypeCode *)&DDS_g_tc_long;
    /* Initialize the values for member annotations. */
    HelloWorld_g_tc_members[0]._annotations._defaultValue._d = RTI_XCDR_TK_LONG;
    HelloWorld_g_tc_members[0]._annotations._defaultValue._u.long_value = 0;
    HelloWorld_g_tc_members[0]._annotations._minValue._d = RTI_XCDR_TK_LONG;
    HelloWorld_g_tc_members[0]._annotations._minValue._u.long_value = RTIXCdrLong_MIN;
    HelloWorld_g_tc_members[0]._annotations._maxValue._d = RTI_XCDR_TK_LONG;
    HelloWorld_g_tc_members[0]._annotations._maxValue._u.long_value = RTIXCdrLong_MAX;
    HelloWorld_g_tc._data._sampleAccessInfo =
    HelloWorld_get_sample_access_info();
    HelloWorld_g_tc._data._typePlugin =
    HelloWorld_get_type_plugin_info();
    RTIOsapiAtomic_store32(
        &is_initialized,
        RTI_TRUE,
        RTI_OSAPI_ATOMIC_MEMORY_ORDER_RELEASE);
    return &HelloWorld_g_tc;
}
RTIXCdrSampleAccessInfo *HelloWorld_get_sample_access_info()
{
    static RTI_ATOMIC(RTIBool) is_initialized;
    static RTIXCdrMemberAccessInfo HelloWorld_g_memberAccessInfos[1] =
    {RTIXCdrMemberAccessInfo_INITIALIZER};
    static RTIXCdrSampleAccessInfo HelloWorld_g_sampleAccessInfo =
    RTIXCdrSampleAccessInfo_INITIALIZER;
    if (RTIOsapiAtomic_load32(
        &is_initialized,
        RTI_OSAPI_ATOMIC_MEMORY_ORDER_ACQUIRE)) {
        return (RTIXCdrSampleAccessInfo*) &HelloWorld_g_sampleAccessInfo;
    }
    HelloWorld_g_memberAccessInfos[0].bindingMemberValueOffset[0] =
    offsetof(struct HelloWorld, data);
    HelloWorld_g_sampleAccessInfo.memberAccessInfos =
    HelloWorld_g_memberAccessInfos;
    {
        size_t candidateTypeSize = sizeof(HelloWorld);
        if (candidateTypeSize > RTIXCdrLong_MAX) {
            HelloWorld_g_sampleAccessInfo.typeSize[0] =
            RTIXCdrLong_MAX;
        } else {
            HelloWorld_g_sampleAccessInfo.typeSize[0] =
            (RTIXCdrUnsignedLong) candidateTypeSize;
        }
    }
    HelloWorld_g_sampleAccessInfo.languageBinding =
    RTI_XCDR_TYPE_BINDING_C ;
    RTIOsapiAtomic_store32(
        &is_initialized,
        RTI_TRUE,
        RTI_OSAPI_ATOMIC_MEMORY_ORDER_RELEASE);
    return (RTIXCdrSampleAccessInfo*) &HelloWorld_g_sampleAccessInfo;
}
RTIXCdrTypePlugin *HelloWorld_get_type_plugin_info()
{
    static RTIXCdrTypePlugin HelloWorld_g_typePlugin =
    {
        NULL, /* serialize */
        NULL, /* serialize_key */
        NULL, /* deserialize_sample */
        NULL, /* deserialize_key_sample */
        NULL, /* skip */
        NULL, /* get_serialized_sample_size */
        NULL, /* get_serialized_sample_max_size_ex */
        NULL, /* get_serialized_key_max_size_ex */
        NULL, /* get_serialized_sample_min_size */
        NULL, /* serialized_sample_to_key */
        (RTIXCdrTypePluginInitializeSampleFunction)
        HelloWorld_initialize_ex,
        NULL,
```

```
        (RTIXCdrTypePluginFinalizeSampleFunction)
        HelloWorld_finalize_w_return,
        NULL,
        NULL
    };
    return &HelloWorld_g_typePlugin;
}
#endif
RTIBool HelloWorld_initialize(
    HelloWorld* sample)
{
    return HelloWorld_initialize_ex(
        sample,
        RTI_TRUE,
        RTI_TRUE);
}
RTIBool HelloWorld_initialize_w_params(
    HelloWorld *sample,
    const struct DDS_TypeAllocationParams_t *allocParams)
{
    if (sample == NULL) {
        return RTI_FALSE;
    }
    if (allocParams == NULL) {
        return RTI_FALSE;
    }
    sample->data = 0;
    return RTI_TRUE;
}
RTIBool HelloWorld_initialize_ex(
    HelloWorld *sample,
    RTIBool allocatePointers,
    RTIBool allocateMemory)
{
    struct DDS_TypeAllocationParams_t allocParams =
    DDS_TYPE_ALLOCATION_PARAMS_DEFAULT;
    allocParams.allocate_pointers =  (DDS_Boolean)allocatePointers;
    allocParams.allocate_memory = (DDS_Boolean)allocateMemory;
    return HelloWorld_initialize_w_params(
        sample,
        &allocParams);
}
RTIBool HelloWorld_finalize_w_return(
    HelloWorld* sample)
{
    HelloWorld_finalize_ex(sample, RTI_TRUE);
    return RTI_TRUE;
}
void HelloWorld_finalize(
    HelloWorld* sample)
{
    HelloWorld_finalize_ex(
        sample,
        RTI_TRUE);
}
void HelloWorld_finalize_ex(
    HelloWorld *sample,
    RTIBool deletePointers)
{
    struct DDS_TypeDeallocationParams_t deallocParams =
    DDS_TYPE_DEALLOCATION_PARAMS_DEFAULT;
    if (sample==NULL) {
        return;
    }
    deallocParams.delete_pointers = (DDS_Boolean)deletePointers;
    HelloWorld_finalize_w_params(
        sample,
        &deallocParams);
}
void HelloWorld_finalize_w_params(
    HelloWorld *sample,
    const struct DDS_TypeDeallocationParams_t *deallocParams)
{
    if (sample==NULL) {
        return;
    }
    if (deallocParams == NULL) {
        return;
    }
}
void HelloWorld_finalize_optional_members(
```

```
    HelloWorld* sample, RTIBool deletePointers)
{
    struct DDS_TypeDeallocationParams_t deallocParamsTmp =
    DDS_TYPE_DEALLOCATION_PARAMS_DEFAULT;
    struct DDS_TypeDeallocationParams_t * deallocParams =
    &deallocParamsTmp;
    if (sample==NULL) {
        return;
    }
    RTIOsapiUtility_unusedParameter(deallocParams);
    deallocParamsTmp.delete_pointers = (DDS_Boolean)deletePointers;
    deallocParamsTmp.delete_optional_members = DDS_BOOLEAN_TRUE;
}
RTIBool HelloWorld_copy(
    HelloWorld* dst,
    const HelloWorld* src)
{
    if (dst == NULL || src == NULL) {
        return RTI_FALSE;
    }
    if (!RTICdrType_copyLong (
        &dst->data,
        &src->data)) {
        return RTI_FALSE;
    }
    return RTI_TRUE;
}
#define T HelloWorld
#define TSeq HelloWorldSeq
#define T_initialize_w_params HelloWorld_initialize_w_params
#define T_finalize_w_params   HelloWorld_finalize_w_params
#define T_copy       HelloWorld_copy
#ifndef NDDS_STANDALONE_TYPE
#include "dds_c/generic/dds_c_sequence_TSeq.gen"
#else
#include "dds_c_sequence_TSeq.gen"
#endif
#undef T_copy
#undef T_finalize_w_params
#undef T_initialize_w_params
#undef TSeq
#undef T
```

5.7 HelloWorld_publisher.c

[$(NDDSHOME)/example/C/helloWorldPersistence/HelloWorld_publisher.c]

```
/* HelloWorld_publisher.c

   A publication of data of type HelloWorld

   This file is derived from code automatically generated by the rtiddsgen
   command:

   rtiddsgen -language C -example <arch> HelloWorld.idl

modification history
------------ -------
*/
#include <stdio.h>
#include <stdlib.h>
#include "ndds/ndds_c.h"
#include "HelloWorld.h"
#include "HelloWorldSupport.h"
/* Delete all entities */
static int publisher_shutdown(
    DDS_DomainParticipant *participant)
{
    DDS_ReturnCode_t retcode;
    int status = 0;
    if (participant != NULL) {
        retcode = DDS_DomainParticipant_delete_contained_entities(participant);
        if (retcode != DDS_RETCODE_OK) {
            printf("delete_contained_entities error %d\n", retcode);
            status = -1;
        }
```

```
        retcode = DDS_DomainParticipantFactory_delete_participant(
            DDS_TheParticipantFactory, participant);
        if (retcode != DDS_RETCODE_OK) {
            printf("delete_participant error %d\n", retcode);
            status = -1;
        }
    }
    /* RTI Connext provides finalize_instance() method for
       people who want to release memory used by the participant factory
       singleton. Uncomment the following block of code for clean destruction of
       the participant factory singleton. */
/*
    retcode = DDS_DomainParticipantFactory_finalize_instance();
    if (retcode != DDS_RETCODE_OK) {
        printf("finalize_instance error %d\n", retcode);
        status = -1;
    }
*/
    return status;
}
int publisher_main(int domainId, int sample_count)
{
    DDS_DomainParticipant *participant = NULL;
    DDS_Publisher *publisher = NULL;
    DDS_Topic *topic = NULL;
    DDS_DataWriter *writer = NULL;
    HelloWorldDataWriter *HelloWorld_writer = NULL;
    HelloWorld *instance = NULL;
    DDS_ReturnCode_t retcode;
    DDS_InstanceHandle_t instance_handle = DDS_HANDLE_NIL;
    const char *type_name = NULL;
    int count = 0;
    struct DDS_Duration_t send_period = {4,0};
    /* To customize participant QoS, use
       DDS_DomainParticipantFactory_get_default_participant_qos() */
    participant = DDS_DomainParticipantFactory_create_participant(
        DDS_TheParticipantFactory, domainId, &DDS_PARTICIPANT_QOS_DEFAULT,
        NULL /* listener */, DDS_STATUS_MASK_NONE);
    if (participant == NULL) {
        printf("create_participant error\n");
        publisher_shutdown(participant);
        return -1;
    }
    /* To customize publisher QoS, use
       DDS_DomainParticipant_get_default_publisher_qos() */
    publisher = DDS_DomainParticipant_create_publisher(
        participant, &DDS_PUBLISHER_QOS_DEFAULT, NULL /* listener */,
        DDS_STATUS_MASK_NONE);
    if (publisher == NULL) {
        printf("create_publisher error\n");
        publisher_shutdown(participant);
        return -1;
    }
    /* Register type before creating topic */
    type_name = HelloWorldTypeSupport_get_type_name();
    retcode = HelloWorldTypeSupport_register_type(
        participant, type_name);
    if (retcode != DDS_RETCODE_OK) {
        printf("register_type error %d\n", retcode);
        publisher_shutdown(participant);
        return -1;
    }
    /* To customize topic QoS, use
       DDS_DomainParticipant_get_default_topic_qos() */
    topic = DDS_DomainParticipant_create_topic(
        participant, "Example HelloWorld",
        type_name, &DDS_TOPIC_QOS_DEFAULT, NULL /* listener */,
        DDS_STATUS_MASK_NONE);
    if (topic == NULL) {
        printf("create_topic error\n");
        publisher_shutdown(participant);
        return -1;
    }
    /* Create data writer */
    writer = DDS_Publisher_create_datawriter(
        publisher, topic,
        &DDS_DATAWRITER_QOS_DEFAULT, NULL /* listener */, DDS_STATUS_MASK_NONE);
    if (writer == NULL) {
        printf("create_datawriter error\n");
        publisher_shutdown(participant);
        return -1;
```

```
    }
    HelloWorld_writer = HelloWorldDataWriter_narrow(writer);
    if (HelloWorld_writer == NULL) {
        printf("DataWriter narrow error\n");
        publisher_shutdown(participant);
        return -1;
    }
    /* Create data sample for writing */
    instance = HelloWorldTypeSupport_create_data_ex(DDS_BOOLEAN_TRUE);
    if (instance == NULL) {
        printf("HelloWorldTypeSupport_create_data error\n");
        publisher_shutdown(participant);
        return -1;
    }
    /* For data type that has key, if the same instance is going to be
       written multiple times, initialize the key here
       and register the keyed instance prior to writing */
/*
    instance_handle = HelloWorldDataWriter_register_instance(
        HelloWorld_writer, instance);
*/
    /* Main loop */
    for (count=0; (sample_count == 0) || (count < sample_count); ++count) {
        printf("Writing HelloWorld, count %d\n", count);
        fflush(stdout);
        /* Modify the data to be written here */
        instance->data = count;
        /* Write data */
        retcode = HelloWorldDataWriter_write(
            HelloWorld_writer, instance, &instance_handle);
        if (retcode != DDS_RETCODE_OK) {
            printf("write error %d\n", retcode);
        }
        NDDS_Utility_sleep(&send_period);
    }
/*
    retcode = HelloWorldDataWriter_unregister_instance(
        HelloWorld_writer, instance, &instance_handle);
    if (retcode != DDS_RETCODE_OK) {
        printf("unregister instance error %d\n", retcode);
    }
*/
    /* Delete data sample */
    retcode = HelloWorldTypeSupport_delete_data_ex(instance, DDS_BOOLEAN_TRUE);
    if (retcode != DDS_RETCODE_OK) {
        printf("HelloWorldTypeSupport_delete_data error %d\n", retcode);
    }

    /* Cleanup and delete delete all entities */
    return publisher_shutdown(participant);
}
#if defined(RTI_WINCE)
int wmain(int argc, wchar_t** argv)
{
    int domainId = 0;
    int sample_count = 0; /* infinite loop */

    if (argc >= 2) {
        domainId = _wtoi(argv[1]);
    }
    if (argc >= 3) {
        sample_count = _wtoi(argv[2]);
    }
    /* Uncomment this to turn on additional logging
    NDDS_Config_Logger_set_verbosity_by_category(
        NDDS_Config_Logger_get_instance(),
        NDDS_CONFIG_LOG_CATEGORY_API,
        NDDS_CONFIG_LOG_VERBOSITY_STATUS_ALL);
    */

    return publisher_main(domainId, sample_count);
}
#elif !(defined(RTI_VXWORKS) && !defined(__RTP__)) && !defined(RTI_PSOS)
int main(int argc, char *argv[])
{
    int domainId = 0;
    int sample_count = 0; /* infinite loop */
    if (argc >= 2) {
        domainId = atoi(argv[1]);
    }
    if (argc >= 3) {
```

```
        sample_count = atoi(argv[2]);
    }

    /* Uncomment this to turn on additional logging
    NDDS_Config_Logger_set_verbosity_by_category(
        NDDS_Config_Logger_get_instance(),
        NDDS_CONFIG_LOG_CATEGORY_API,
        NDDS_CONFIG_LOG_VERBOSITY_STATUS_ALL);
    */

    return publisher_main(domainId, sample_count);
}
#endif
```

5.8 HelloWorld_subscriber.c

[$(NDDSHOME)/example/C/helloWorldPersistence/HelloWorld_subscriber.c]

```
/* HelloWorld_subscriber.c

   A subscription example

   This file is derived from code automatically generated by the rtiddsgen
   command:

   rtiddsgen -language C -example <arch> HelloWorld.idl

modification history
------------ -------
*/
#include <stdio.h>
#include <stdlib.h>
#include "ndds/ndds_c.h"
#include "HelloWorld.h"
#include "HelloWorldSupport.h"
void HelloWorldListener_on_requested_deadline_missed(
    void* listener_data,
    DDS_DataReader* reader,
    const struct DDS_RequestedDeadlineMissedStatus *status)
{
}
void HelloWorldListener_on_requested_incompatible_qos(
    void* listener_data,
    DDS_DataReader* reader,
    const struct DDS_RequestedIncompatibleQosStatus *status)
{
}
void HelloWorldListener_on_sample_rejected(
    void* listener_data,
    DDS_DataReader* reader,
    const struct DDS_SampleRejectedStatus *status)
{
}
void HelloWorldListener_on_liveliness_changed(
    void* listener_data,
    DDS_DataReader* reader,
    const struct DDS_LivelinessChangedStatus *status)
{
}
void HelloWorldListener_on_sample_lost(
    void* listener_data,
    DDS_DataReader* reader,
    const struct DDS_SampleLostStatus *status)
{
}
void HelloWorldListener_on_subscription_matched(
    void* listener_data,
    DDS_DataReader* reader,
    const struct DDS_SubscriptionMatchedStatus *status)
{
}
void HelloWorldListener_on_data_available(
    void* listener_data,
    DDS_DataReader* reader)
{
    HelloWorldDataReader *HelloWorld_reader = NULL;
```

```
    struct HelloWorldSeq data_seq = DDS_SEQUENCE_INITIALIZER;
    struct DDS_SampleInfoSeq info_seq = DDS_SEQUENCE_INITIALIZER;
    DDS_ReturnCode_t retcode;
    int i;
    HelloWorld_reader = HelloWorldDataReader_narrow(reader);
    if (HelloWorld_reader == NULL) {
        printf("DataReader narrow error\n");
        return;
    }
    retcode = HelloWorldDataReader_take(
        HelloWorld_reader,
        &data_seq, &info_seq, DDS_LENGTH_UNLIMITED,
        DDS_ANY_SAMPLE_STATE, DDS_ANY_VIEW_STATE, DDS_ANY_INSTANCE_STATE);
    if (retcode == DDS_RETCODE_NO_DATA) {
        return;
    } else if (retcode != DDS_RETCODE_OK) {
        printf("take error %d\n", retcode);
        return;
    }
    for (i = 0; i < HelloWorldSeq_get_length(&data_seq); ++i) {
        if (DDS_SampleInfoSeq_get_reference(&info_seq, i)->valid_data) {
            HelloWorldTypeSupport_print_data(
                HelloWorldSeq_get_reference(&data_seq, i));
        }
    }
    retcode = HelloWorldDataReader_return_loan(
        HelloWorld_reader,
        &data_seq, &info_seq);
    if (retcode != DDS_RETCODE_OK) {
        printf("return loan error %d\n", retcode);
    }
}
/* Delete all entities */
static int subscriber_shutdown(
    DDS_DomainParticipant *participant)
{
    DDS_ReturnCode_t retcode;
    int status = 0;
    if (participant != NULL) {
        retcode = DDS_DomainParticipant_delete_contained_entities(participant);
        if (retcode != DDS_RETCODE_OK) {
            printf("delete_contained_entities error %d\n", retcode);
            status = -1;
        }
        retcode = DDS_DomainParticipantFactory_delete_participant(
            DDS_TheParticipantFactory, participant);
        if (retcode != DDS_RETCODE_OK) {
            printf("delete_participant error %d\n", retcode);
            status = -1;
        }
    }
    /* RTI Connext provides finalize_instance() method for
       people who want to release memory used by the participant factory
       singleton. Uncomment the following block of code for clean destruction of
       the participant factory singleton. */
/*
    retcode = DDS_DomainParticipantFactory_finalize_instance();
    if (retcode != DDS_RETCODE_OK) {
        printf("finalize_instance error %d\n", retcode);
        status = -1;
    }
*/
    return status;
}
int subscriber_main(int domainId, int sample_count)
{
    DDS_DomainParticipant *participant = NULL;
    DDS_Subscriber *subscriber = NULL;
    DDS_Topic *topic = NULL;
    struct DDS_DataReaderListener reader_listener =
        DDS_DataReaderListener_INITIALIZER;
    DDS_DataReader *reader = NULL;
    DDS_ReturnCode_t retcode;
    const char *type_name = NULL;
    int count = 0;
    struct DDS_Duration_t poll_period = {4,0};
    /* To customize participant QoS, use
       DDS_DomainParticipantFactory_get_default_participant_qos() */
    participant = DDS_DomainParticipantFactory_create_participant(
        DDS_TheParticipantFactory, domainId, &DDS_PARTICIPANT_QOS_DEFAULT,
        NULL /* listener */, DDS_STATUS_MASK_NONE);
```

```
    if (participant == NULL) {
        printf("create_participant error\n");
        subscriber_shutdown(participant);
        return -1;
    }
    /* To customize subscriber QoS, use
       DDS_DomainParticipant_get_default_datarea_qos() */
    subscriber = DDS_DomainParticipant_create_subscriber(
        participant, &DDS_SUBSCRIBER_QOS_DEFAULT, NULL /* listener */,
        DDS_STATUS_MASK_NONE);
    if (subscriber == NULL) {
        printf("create_subscriber error\n");
        subscriber_shutdown(participant);
        return -1;
    }
    /* Register type before creating topic */
    type_name = HelloWorldTypeSupport_get_type_name();
    retcode = HelloWorldTypeSupport_register_type(participant, type_name);
    if (retcode != DDS_RETCODE_OK) {
        printf("register_type error %d\n", retcode);
        subscriber_shutdown(participant);
        return -1;
    }
    /* To customize topic QoS, use
       DDS_DomainParticipant_get_default_topic_qos() */
    topic = DDS_DomainParticipant_create_topic(
        participant, "Example HelloWorld",
        type_name, &DDS_TOPIC_QOS_DEFAULT, NULL /* listener */,
        DDS_STATUS_MASK_NONE);
    if (topic == NULL) {
        printf("create_topic error\n");
        subscriber_shutdown(participant);
        return -1;
    }
    /* Setup data reader listener */
    reader_listener.on_requested_deadline_missed  =
        HelloWorldListener_on_requested_deadline_missed;
    reader_listener.on_requested_incompatible_qos =
        HelloWorldListener_on_requested_incompatible_qos;
    reader_listener.on_sample_rejected =
        HelloWorldListener_on_sample_rejected;
    reader_listener.on_liveliness_changed =
        HelloWorldListener_on_liveliness_changed;
    reader_listener.on_sample_lost =
        HelloWorldListener_on_sample_lost;
    reader_listener.on_subscription_matched =
        HelloWorldListener_on_subscription_matched;
    reader_listener.on_data_available =
        HelloWorldListener_on_data_available;
    /* Create data reader */
    reader = DDS_Subscriber_create_datareader(
        subscriber, DDS_Topic_as_topicdescription(topic),
        &DDS_DATAREADER_QOS_DEFAULT, &reader_listener, DDS_STATUS_MASK_ALL);
    if (reader == NULL) {
        printf("create_datareader error\n");
        subscriber_shutdown(participant);
        return -1;
    }
    /* Main loop */
    for (count=0; (sample_count == 0) || (count < sample_count); ++count) {
        printf("HelloWorld subscriber sleeping for %d sec...\n",
               poll_period.sec);
        fflush(stdout);
        NDDS_Utility_sleep(&poll_period);
    }
    /* Cleanup and delete delete all entities */
    return subscriber_shutdown(participant);
}
#if defined(RTI_WINCE)
int wmain(int argc, wchar_t** argv)
{
    int domainId = 0;
    int sample_count = 0; /* infinite loop */

    if (argc >= 2) {
        domainId = _wtoi(argv[1]);
    }
    if (argc >= 3) {
        sample_count = _wtoi(argv[2]);
    }
    /* Uncomment this to turn on additional logging
```

```
    NDDS_Config_Logger_set_verbosity_by_category(
        NDDS_Config_Logger_get_instance(),
        NDDS_CONFIG_LOG_CATEGORY_API,
        NDDS_CONFIG_LOG_VERBOSITY_STATUS_ALL);
    */

    return subscriber_main(domainId, sample_count);
}
#elif !(defined(RTI_VXWORKS) && !defined(__RTP__)) && !defined(RTI_PSOS)
int main(int argc, char *argv[])
{
    int domainId = 0;
    int sample_count = 0; /* infinite loop */
    if (argc >= 2) {
        domainId = atoi(argv[1]);
    }
    if (argc >= 3) {
        sample_count = atoi(argv[2]);
    }
    /* Uncomment this to turn on additional logging
    NDDS_Config_Logger_set_verbosity_by_category(
        NDDS_Config_Logger_get_instance(),
        NDDS_CONFIG_LOG_CATEGORY_API,
        NDDS_CONFIG_LOG_VERBOSITY_STATUS_ALL);
    */

    return subscriber_main(domainId, sample_count);
}
#endif
```

5.9 HelloWorldPlugin.c

[$(NDDSHOME)/example/C/helloWorldPersistence/HelloWorldPlugin.h]

```
/*
WARNING: THIS FILE IS AUTO-GENERATED. DO NOT MODIFY.

This file was generated from HelloWorld.idl
using RTI Code Generator (rtiddsgen) version 4.7.0.
The rtiddsgen tool is part of the RTI Connext DDS distribution.
For more information, type 'rtiddsgen -help' at a command shell
or consult the Code Generator User's Manual.
*/
#ifndef HelloWorldPlugin_1436885481_h
#define HelloWorldPlugin_1436885481_h
#include "HelloWorld.h"
struct RTICdrStream;
#ifndef pres_typePlugin_h
#include "pres/pres_typePlugin.h"
#endif
#if defined(NDDS_USER_DLL_EXPORT) && defined(RTI_WIN32)
#undef NDDSUSERDllExport
#define NDDSUSERDllExport __declspec(dllexport)
#endif
#if !defined(RTI_WIN32) && defined(NDDS_USER_SYMBOL_EXPORT)
#undef NDDSUSERDllExport
#define NDDSUSERDllExport __attribute__((visibility("default")))
#endif
#ifdef __cplusplus
extern "C" {
    #endif
    #define HelloWorldPlugin_get_sample PRESTypePluginDefaultEndpointData_getSample
    #define HelloWorldPlugin_get_buffer PRESTypePluginDefaultEndpointData_getBuffer
    #define HelloWorldPlugin_return_buffer PRESTypePluginDefaultEndpointData_returnBuffer
    #define HelloWorldPlugin_create_sample PRESTypePluginDefaultEndpointData_createSample
    #define HelloWorldPlugin_destroy_sample PRESTypePluginDefaultEndpointData_deleteSample
    /* --------------------------------------------------------------------------------------
    Support functions:
    * -------------------------------------------------------------------------------------- */
    NDDSUSERDllExport extern HelloWorld*
    HelloWorldPluginSupport_create_data_w_params(
        const struct DDS_TypeAllocationParams_t * alloc_params);
    NDDSUSERDllExport extern HelloWorld*
    HelloWorldPluginSupport_create_data_ex(RTIBool allocate_pointers);
    NDDSUSERDllExport extern HelloWorld*
    HelloWorldPluginSupport_create_data(void);
```

```
NDDSUSERDllExport extern RTIBool
HelloWorldPluginSupport_copy_data(
    HelloWorld *out,
    const HelloWorld *in);
NDDSUSERDllExport extern void
HelloWorldPluginSupport_destroy_data_w_params(
    HelloWorld *sample,
    const struct DDS_TypeDeallocationParams_t * dealloc_params);
NDDSUSERDllExport extern void
HelloWorldPluginSupport_destroy_data_ex(
    HelloWorld *sample,RTIBool deallocate_pointers);
NDDSUSERDllExport extern void
HelloWorldPluginSupport_destroy_data(
    HelloWorld *sample);
NDDSUSERDllExport extern void
HelloWorldPluginSupport_print_data(
    const HelloWorld *sample,
    const char *desc,
    unsigned int indent);
/* ----------------------------------------------------------------------------
Callback functions:
* ---------------------------------------------------------------------------- */
NDDSUSERDllExport extern PRESTypePluginParticipantData
HelloWorldPlugin_on_participant_attached(
    void *registration_data,
    const struct PRESTypePluginParticipantInfo *participant_info,
    RTIBool top_level_registration,
    void *container_plugin_context,
    RTICdrTypeCode *typeCode);
NDDSUSERDllExport extern void
HelloWorldPlugin_on_participant_detached(
    PRESTypePluginParticipantData participant_data);
NDDSUSERDllExport extern PRESTypePluginEndpointData
HelloWorldPlugin_on_endpoint_attached(
    PRESTypePluginParticipantData participant_data,
    const struct PRESTypePluginEndpointInfo *endpoint_info,
    RTIBool top_level_registration,
    void *container_plugin_context);
NDDSUSERDllExport extern void
HelloWorldPlugin_on_endpoint_detached(
    PRESTypePluginEndpointData endpoint_data);
NDDSUSERDllExport extern void
HelloWorldPlugin_return_sample(
    PRESTypePluginEndpointData endpoint_data,
    HelloWorld *sample,
    void *handle);
NDDSUSERDllExport extern RTIBool
HelloWorldPlugin_copy_sample(
    PRESTypePluginEndpointData endpoint_data,
    HelloWorld *out,
    const HelloWorld *in);
/* ----------------------------------------------------------------------------
(De)Serialize functions:
* ---------------------------------------------------------------------------- */
NDDSUSERDllExport extern RTIBool
HelloWorldPlugin_serialize_to_cdr_buffer(
    char * buffer,
    unsigned int * length,
    const HelloWorld *sample);
NDDSUSERDllExport extern RTIBool
HelloWorldPlugin_serialize_to_cdr_buffer_ex(
    char *buffer,
    unsigned int *length,
    const HelloWorld *sample,
    DDS_DataRepresentationId_t representation);
NDDSUSERDllExport extern RTIBool
HelloWorldPlugin_deserialize_from_cdr_buffer(
    HelloWorld *sample,
    const char * buffer,
    unsigned int length);
#if !defined (NDDS_STANDALONE_TYPE)
NDDSUSERDllExport extern DDS_ReturnCode_t
HelloWorldPlugin_data_to_string(
    const HelloWorld *sample,
    char *str,
    DDS_UnsignedLong *str_size,
    const struct DDS_PrintFormatProperty *property);
#endif
NDDSUSERDllExport extern unsigned int
HelloWorldPlugin_get_serialized_sample_max_size(
    PRESTypePluginEndpointData endpoint_data,
```

```
        RTIBool include_encapsulation,
        RTIEncapsulationId encapsulation_id,
        unsigned int current_alignment);
    /* ---------------------------------------------------------------------------------------
    Key Management functions:
    * --------------------------------------------------------------------------------------- */
    NDDSUSERDllExport extern PRESTypePluginKeyKind
    HelloWorldPlugin_get_key_kind(void);
    NDDSUSERDllExport extern unsigned int
    HelloWorldPlugin_get_serialized_key_max_size(
        PRESTypePluginEndpointData endpoint_data,
        RTIBool include_encapsulation,
        RTIEncapsulationId encapsulation_id,
        unsigned int current_alignment);
    NDDSUSERDllExport extern unsigned int
    HelloWorldPlugin_get_serialized_key_max_size_for_keyhash(
        PRESTypePluginEndpointData endpoint_data,
        RTIEncapsulationId encapsulation_id,
        unsigned int current_alignment);
    NDDSUSERDllExport extern RTIBool
    HelloWorldPlugin_deserialize_key(
        PRESTypePluginEndpointData endpoint_data,
        HelloWorld ** sample,
        RTIBool * drop_sample,
        struct RTICdrStream *cdrStream,
        RTIBool deserialize_encapsulation,
        RTIBool deserialize_key,
        void *endpoint_plugin_qos);
    NDDSUSERDllExport extern
    struct RTIXCdrInterpreterPrograms * HelloWorldPlugin_get_programs(void);
    /* Plugin Functions */
    NDDSUSERDllExport extern struct PRESTypePlugin*
    HelloWorldPlugin_new(void);
    NDDSUSERDllExport extern void
    HelloWorldPlugin_delete(struct PRESTypePlugin *);
    #ifdef __cplusplus
}
#endif
#if defined(NDDS_USER_DLL_EXPORT) || defined(NDDS_USER_SYMBOL_EXPORT)
#undef NDDSUSERDllExport
#define NDDSUSERDllExport
#endif
#endif /* HelloWorldPlugin_1436885481_h */
```

[$(NDDSHOME)/example/C/helloWorldPersistence/HelloWorldPlugin.c]

```
/*
WARNING: THIS FILE IS AUTO-GENERATED. DO NOT MODIFY.

This file was generated from HelloWorld.idl
using RTI Code Generator (rtiddsgen) version 4.7.0.
The rtiddsgen tool is part of the RTI Connext DDS distribution.
For more information, type 'rtiddsgen -help' at a command shell
or consult the Code Generator User's Manual.
*/
#include <string.h>
#ifndef ndds_c_h
#include "ndds/ndds_c.h"
#endif
#ifndef cdr_type_h
#include "cdr/cdr_type.h"
#endif
#ifndef cdr_type_object_h
#include "cdr/cdr_typeObject.h"
#endif
#ifndef cdr_encapsulation_h
#include "cdr/cdr_encapsulation.h"
#endif
#ifndef cdr_stream_h
#include "cdr/cdr_stream.h"
#endif
#include "xcdr/xcdr_interpreter.h"
#include "xcdr/xcdr_stream.h"
#ifndef cdr_log_h
#include "cdr/cdr_log.h"
#endif
#ifndef pres_typePlugin_h
#include "pres/pres_typePlugin.h"
#endif
#include "dds_c/dds_c_typecode_impl.h"
#define RTI_CDR_CURRENT_SUBMODULE RTI_CDR_SUBMODULE_MASK_STREAM
```

```
#include "HelloWorldPlugin.h"
/* ------------------------------------------------------------------------
*  Type HelloWorld
* ------------------------------------------------------------------------ */
/* ------------------------------------------------------------------------
Support functions:
* ------------------------------------------------------------------------ */
HelloWorld*
HelloWorldPluginSupport_create_data_w_params(
    const struct DDS_TypeAllocationParams_t * alloc_params)
{
    HelloWorld *sample = NULL;
    if (alloc_params == NULL) {
        return NULL;
    } else if(!alloc_params->allocate_memory) {
        RTICdrLog_exception(&RTI_CDR_LOG_TYPE_OBJECT_NOT_ASSIGNABLE_ss,
        "alloc_params->allocate_memory","false");
        return NULL;
    }
    RTIOsapiHeap_allocateStructure(&(sample),HelloWorld);
    if (sample == NULL) {
        return NULL;
    }
    if (!HelloWorld_initialize_w_params(sample,alloc_params)) {
        struct DDS_TypeDeallocationParams_t deallocParams =
        DDS_TYPE_DEALLOCATION_PARAMS_DEFAULT;
        deallocParams.delete_pointers = alloc_params->allocate_pointers;
        deallocParams.delete_optional_members = alloc_params->allocate_pointers;
        /* Coverity reports a possible uninit_use_in_call that will happen if the
        allocation fails. But if the allocation fails then sample == null and
        the method will return before reach this point.*/
        /* Coverity reports a possible overwrite_var on the members of the sample.
        It is a false positive since all the pointers are freed before assigning
        null to them. */
        /* coverity[uninit_use_in_call : FALSE] */
        /* coverity[overwrite_var : FALSE] */
        HelloWorld_finalize_w_params(sample, &deallocParams);
        /* Coverity reports a possible leaked_storage on the sample members when
        freeing sample. It is a false positive since all the members' memory
        is freed in the call "HelloWorld_finalize_ex" */
        /* coverity[leaked_storage : FALSE] */
        RTIOsapiHeap_freeStructure(sample);
        sample=NULL;
    }
    return sample;
}
HelloWorld *
HelloWorldPluginSupport_create_data_ex(RTIBool allocate_pointers)
{
    HelloWorld *sample = NULL;
    RTIOsapiHeap_allocateStructure(&(sample),HelloWorld);
    if(sample == NULL) {
        return NULL;
    }
    /* coverity[example_checked : FALSE] */
    if (!HelloWorld_initialize_ex(sample,allocate_pointers, RTI_TRUE)) {
        /* Coverity reports a possible uninit_use_in_call that will happen if the
        new fails. But if new fails then sample == null and the method will
        return before reach this point. */
        /* Coverity reports a possible overwrite_var on the members of the sample.
        It is a false positive since all the pointers are freed before assigning
        null to them. */
        /* coverity[uninit_use_in_call : FALSE] */
        /* coverity[overwrite_var : FALSE] */
        HelloWorld_finalize_ex(sample, RTI_TRUE);
        /* Coverity reports a possible leaked_storage on the sample members when
        freeing sample. It is a false positive since all the members' memory
        is freed in the call "HelloWorld_finalize_ex" */
        /* coverity[leaked_storage : FALSE] */
        RTIOsapiHeap_freeStructure(sample);
        sample=NULL;
    }
    return sample;
}
void *
HelloWorldPluginSupport_create_dataI(void)
{
    return HelloWorldPluginSupport_create_data_ex(RTI_TRUE);
}
HelloWorld *
HelloWorldPluginSupport_create_data(void)
```

```
{
    return (HelloWorld *) HelloWorldPluginSupport_create_dataI();
}
void
HelloWorldPluginSupport_destroy_data_w_params(
    HelloWorld *sample,
    const struct DDS_TypeDeallocationParams_t * dealloc_params) {
    HelloWorld_finalize_w_params(sample,dealloc_params);
    RTIOsapiHeap_freeStructure(sample);
}
void
HelloWorldPluginSupport_destroy_data_ex(
    HelloWorld *sample,RTIBool deallocate_pointers) {
    HelloWorld_finalize_ex(sample,deallocate_pointers);
    RTIOsapiHeap_freeStructure(sample);
}
void
HelloWorldPluginSupport_destroy_data(
    HelloWorld *sample) {
    HelloWorldPluginSupport_destroy_data_ex(sample,RTI_TRUE);
}
void
HelloWorldPluginSupport_destroy_dataI(
    void *sample)
{
    HelloWorldPluginSupport_destroy_data((HelloWorld *) sample);
}
RTIBool
HelloWorldPluginSupport_copy_data(
    HelloWorld *dst,
    const HelloWorld *src)
{
    return HelloWorld_copy(dst,(const HelloWorld*) src);
}
void
HelloWorldPluginSupport_print_data(
    const HelloWorld *sample,
    const char *desc,
    unsigned int indent_level)
{
    RTICdrType_printIndent(indent_level);
    if (desc != NULL) {
        RTILogParamString_printPlain("%s:\n", desc);
    } else {
        RTILogParamString_printPlain("\n");
    }
    if (sample == NULL) {
        RTILogParamString_printPlain("NULL\n");
        return;
    }
    RTICdrType_printLong(
        &sample->data,
        "data",
        RTIOsapiUtility_uInt32Plus1(indent_level));
}
/* ------------------------------------------------------------------------
Callback functions:
* ------------------------------------------------------------------------ */
PRESTypePluginParticipantData
HelloWorldPlugin_on_participant_attached(
    void *registration_data,
    const struct PRESTypePluginParticipantInfo *participant_info,
    RTIBool top_level_registration,
    void *container_plugin_context,
    RTICdrTypeCode *type_code)
{
    struct RTIXCdrInterpreterPrograms *programs = NULL;
    struct RTIXCdrInterpreterProgramsGenProperty programProperty =
    RTIXCdrInterpreterProgramsGenProperty_INITIALIZER;
    struct PRESTypePluginDefaultParticipantData *pd = NULL;
    RTIOsapiUtility_unusedParameter(registration_data);
    RTIOsapiUtility_unusedParameter(participant_info);
    RTIOsapiUtility_unusedParameter(top_level_registration);
    RTIOsapiUtility_unusedParameter(container_plugin_context);
    RTIOsapiUtility_unusedParameter(type_code);
    pd = (struct PRESTypePluginDefaultParticipantData *)
    PRESTypePluginDefaultParticipantData_new(participant_info);
    programProperty.generateV1Encapsulation = RTI_XCDR_TRUE;
    programProperty.generateV2Encapsulation = RTI_XCDR_TRUE;
    programProperty.resolveAlias = RTI_XCDR_TRUE;
    programProperty.inlineStruct = RTI_XCDR_TRUE;
```

```
    programProperty.optimizeEnum = RTI_XCDR_TRUE;
    programProperty.unboundedSize = RTIXCdrLong_MAX;
    programs = DDS_TypeCodeFactory_assert_programs_in_global_list(
        DDS_TypeCodeFactory_get_instance(),
        HelloWorld_get_typecode(),
        &programProperty,
        RTI_XCDR_PROGRAM_MASK_TYPEPLUGIN);
    if (programs == NULL) {
        PRESTypePluginDefaultParticipantData_delete(
            (PRESTypePluginParticipantData) pd);
        return NULL;
    }
    pd->programs = programs;
    return (PRESTypePluginParticipantData)pd;
}
void
HelloWorldPlugin_on_participant_detached(
    PRESTypePluginParticipantData participant_data)
{
    if (participant_data != NULL) {
        struct PRESTypePluginDefaultParticipantData *pd =
        (struct PRESTypePluginDefaultParticipantData *)participant_data;
        if (pd->programs != NULL) {
            DDS_TypeCodeFactory_remove_programs_from_global_list(
                DDS_TypeCodeFactory_get_instance(),
                pd->programs);
            pd->programs = NULL;
        }
        PRESTypePluginDefaultParticipantData_delete(participant_data);
    }
}
PRESTypePluginEndpointData
HelloWorldPlugin_on_endpoint_attached(
    PRESTypePluginParticipantData participant_data,
    const struct PRESTypePluginEndpointInfo *endpoint_info,
    RTIBool top_level_registration,
    void *containerPluginContext)
{
    PRESTypePluginEndpointData epd = NULL;
    unsigned int serializedSampleMaxSize = 0;
    RTIOsapiUtility_unusedParameter(top_level_registration);
    RTIOsapiUtility_unusedParameter(containerPluginContext);
    if (participant_data == NULL) {
        return NULL;
    }
    epd = PRESTypePluginDefaultEndpointData_new(
        participant_data,
        endpoint_info,
        HelloWorldPluginSupport_create_dataI,
        HelloWorldPluginSupport_destroy_dataI,
        NULL , NULL );
    if (epd == NULL) {
        return NULL;
    }
    if (endpoint_info->endpointKind == PRES_TYPEPLUGIN_ENDPOINT_WRITER) {
        serializedSampleMaxSize = HelloWorldPlugin_get_serialized_sample_max_size(
            epd,RTI_FALSE,RTI_CDR_ENCAPSULATION_ID_CDR_BE,0);
        PRESTypePluginDefaultEndpointData_setMaxSizeSerializedSample(epd, serializedSampleMaxSize);
        if (PRESTypePluginDefaultEndpointData_createWriterPool(
            epd,
            endpoint_info,
            (PRESTypePluginGetSerializedSampleMaxSizeFunction)
            HelloWorldPlugin_get_serialized_sample_max_size, epd,
            (PRESTypePluginGetSerializedSampleSizeFunction)
            PRESTypePlugin_interpretedGetSerializedSampleSize,
            epd) == RTI_FALSE) {
            PRESTypePluginDefaultEndpointData_delete(epd);
            return NULL;
        }
    }
    return epd;
}
void
HelloWorldPlugin_on_endpoint_detached(
    PRESTypePluginEndpointData endpoint_data)
{
    PRESTypePluginDefaultEndpointData_delete(endpoint_data);
}
void
HelloWorldPlugin_return_sample(
    PRESTypePluginEndpointData endpoint_data,
```

```
    HelloWorld *sample,
    void *handle)
{
    HelloWorld_finalize_optional_members(sample, RTI_TRUE);
    PRESTypePluginDefaultEndpointData_returnSample(
        endpoint_data, sample, handle);
}
void HelloWorldPlugin_finalize_optional_members(
    PRESTypePluginEndpointData endpoint_data,
    HelloWorld* sample,
    RTIBool deletePointers)
{
    RTIOsapiUtility_unusedParameter(endpoint_data);
    HelloWorld_finalize_optional_members(
        sample, deletePointers);
}
RTIBool
HelloWorldPlugin_copy_sample(
    PRESTypePluginEndpointData endpoint_data,
    HelloWorld *dst,
    const HelloWorld *src)
{
    RTIOsapiUtility_unusedParameter(endpoint_data);
    return HelloWorldPluginSupport_copy_data(dst,src);
}
/* ------------------------------------------------------------------------
(De)Serialize functions:
* ------------------------------------------------------------------------ */
unsigned int
HelloWorldPlugin_get_serialized_sample_max_size(
    PRESTypePluginEndpointData endpoint_data,
    RTIBool include_encapsulation,
    RTIEncapsulationId encapsulation_id,
    unsigned int current_alignment);
RTIBool
HelloWorldPlugin_serialize_to_cdr_buffer_ex(
    char *buffer,
    unsigned int *length,
    const HelloWorld *sample,
    DDS_DataRepresentationId_t representation)
{
    RTIEncapsulationId encapsulationId = RTI_CDR_ENCAPSULATION_ID_INVALID;
    struct RTICdrStream cdrStream;
    struct PRESTypePluginDefaultEndpointData epd;
    RTIBool result;
    struct PRESTypePluginDefaultParticipantData pd;
    struct RTIXCdrTypePluginProgramContext defaultProgramContext =
    RTIXCdrTypePluginProgramContext_INTIALIZER;
    struct PRESTypePlugin plugin;
    if (length == NULL) {
        return RTI_FALSE;
    }
    RTIOsapiMemory_zero(&epd, sizeof(struct PRESTypePluginDefaultEndpointData));
    epd.programContext = defaultProgramContext;
    epd._participantData = &pd;
    epd.typePlugin = &plugin;
    epd.programContext.endpointPluginData = &epd;
    plugin.typeCode = (struct RTICdrTypeCode *)
    HelloWorld_get_typecode();
    pd.programs = HelloWorldPlugin_get_programs();
    if (pd.programs == NULL) {
        return RTI_FALSE;
    }
    encapsulationId = DDS_TypeCode_get_native_encapsulation(
        (DDS_TypeCode *) plugin.typeCode,
        representation);
    if (encapsulationId == RTI_CDR_ENCAPSULATION_ID_INVALID) {
        return RTI_FALSE;
    }
    epd._maxSizeSerializedSample =
    HelloWorldPlugin_get_serialized_sample_max_size(
        (PRESTypePluginEndpointData)&epd,
        RTI_TRUE,
        encapsulationId,
        0);
    if (buffer == NULL) {
        *length =
        PRESTypePlugin_interpretedGetSerializedSampleSize(
            (PRESTypePluginEndpointData)&epd,
            RTI_TRUE,
            encapsulationId,
```

```
            0,
            sample);
        if (*length == 0) {
            return RTI_FALSE;
        }
        return RTI_TRUE;
    }
    RTICdrStream_init(&cdrStream);
    RTICdrStream_set(&cdrStream, (char *)buffer, *length);
    result = PRESTypePlugin_interpretedSerialize(
        (PRESTypePluginEndpointData)&epd,
        sample,
        &cdrStream,
        RTI_TRUE,
        encapsulationId,
        RTI_TRUE,
        NULL);
    *length = (unsigned int) RTICdrStream_getCurrentPositionOffset(&cdrStream);
    return result;
}
RTIBool
HelloWorldPlugin_serialize_to_cdr_buffer(
    char *buffer,
    unsigned int *length,
    const HelloWorld *sample)
{
    return HelloWorldPlugin_serialize_to_cdr_buffer_ex(
        buffer,
        length,
        sample,
        DDS_AUTO_DATA_REPRESENTATION);
}
RTIBool
HelloWorldPlugin_deserialize_from_cdr_buffer(
    HelloWorld *sample,
    const char * buffer,
    unsigned int length)
{
    struct RTICdrStream cdrStream;
    struct PRESTypePluginDefaultEndpointData epd;
    struct RTIXCdrTypePluginProgramContext defaultProgramContext =
    RTIXCdrTypePluginProgramContext_INTIALIZER;
    struct PRESTypePluginDefaultParticipantData pd;
    struct PRESTypePlugin plugin;
    epd.programContext = defaultProgramContext;
    epd._participantData = &pd;
    epd.typePlugin = &plugin;
    epd.programContext.endpointPluginData = &epd;
    plugin.typeCode = (struct RTICdrTypeCode *)
    HelloWorld_get_typecode();
    pd.programs = HelloWorldPlugin_get_programs();
    if (pd.programs == NULL) {
        return RTI_FALSE;
    }
    RTIXCdrSampleAssignabilityProperty_setFromGlobalComplianceMask(
        &epd._assignabilityProperty);
    RTICdrStream_init(&cdrStream);
    RTICdrStream_set(&cdrStream, (char *)buffer, length);
    HelloWorld_finalize_optional_members(sample, RTI_TRUE);
    return PRESTypePlugin_interpretedDeserialize(
        (PRESTypePluginEndpointData)&epd, sample,
        &cdrStream, RTI_TRUE, RTI_TRUE,
        NULL);
}
#if !defined(NDDS_STANDALONE_TYPE)
DDS_ReturnCode_t
HelloWorldPlugin_data_to_string(
    const HelloWorld *sample,
    char *_str,
    DDS_UnsignedLong *str_size,
    const struct DDS_PrintFormatProperty *property)
{
    DDS_DynamicData *data = NULL;
    char *buffer = NULL;
    unsigned int length = 0;
    struct DDS_PrintFormat printFormat;
    DDS_ReturnCode_t retCode = DDS_RETCODE_ERROR;
    if (sample == NULL) {
        return DDS_RETCODE_BAD_PARAMETER;
    }
    if (str_size == NULL) {
```

```
        return DDS_RETCODE_BAD_PARAMETER;
    }
    if (property == NULL) {
        return DDS_RETCODE_BAD_PARAMETER;
    }
    if (!HelloWorldPlugin_serialize_to_cdr_buffer(
        NULL,
        &length,
        sample)) {
        return DDS_RETCODE_ERROR;
    }
    RTIOsapiHeap_allocateBuffer(&buffer, length, RTI_OSAPI_ALIGNMENT_DEFAULT);
    if (buffer == NULL) {
        return DDS_RETCODE_ERROR;
    }
    if (!HelloWorldPlugin_serialize_to_cdr_buffer(
        buffer,
        &length,
        sample)) {
        RTIOsapiHeap_freeBuffer(buffer);
        return DDS_RETCODE_ERROR;
    }
    data = DDS_DynamicData_new(
        HelloWorld_get_typecode(),
        &DDS_DYNAMIC_DATA_PROPERTY_DEFAULT);
    if (data == NULL) {
        RTIOsapiHeap_freeBuffer(buffer);
        return DDS_RETCODE_ERROR;
    }
    retCode = DDS_DynamicData_from_cdr_buffer(data, buffer, length);
    if (retCode != DDS_RETCODE_OK) {
        RTIOsapiHeap_freeBuffer(buffer);
        DDS_DynamicData_delete(data);
        return retCode;
    }
    retCode = DDS_PrintFormatProperty_to_print_format(
        property,
        &printFormat);
    if (retCode != DDS_RETCODE_OK) {
        RTIOsapiHeap_freeBuffer(buffer);
        DDS_DynamicData_delete(data);
        return retCode;
    }
    retCode = DDS_DynamicDataFormatter_to_string_w_format(
        data,
        _str,
        str_size,
        &printFormat);
    if (retCode != DDS_RETCODE_OK) {
        RTIOsapiHeap_freeBuffer(buffer);
        DDS_DynamicData_delete(data);
        return retCode;
    }
    RTIOsapiHeap_freeBuffer(buffer);
    DDS_DynamicData_delete(data);
    return DDS_RETCODE_OK;
}
#endif
unsigned int
HelloWorldPlugin_get_serialized_sample_max_size(
    PRESTypePluginEndpointData endpoint_data,
    RTIBool include_encapsulation,
    RTIEncapsulationId encapsulation_id,
    unsigned int current_alignment)
{
    unsigned int size;
    RTIBool overflow = RTI_FALSE;
    size = PRESTypePlugin_interpretedGetSerializedSampleMaxSize(
        endpoint_data,&overflow,include_encapsulation,encapsulation_id,current_alignment);
    if (overflow) {
        size = RTI_CDR_MAX_SERIALIZED_SIZE;
    }
    return size;
}
/* ------------------------------------------------------------------------------------
Key Management functions:
* ------------------------------------------------------------------------------------ */
PRESTypePluginKeyKind
HelloWorldPlugin_get_key_kind(void)
{
    return PRES_TYPEPLUGIN_NO_KEY;
```

```
}
RTIBool HelloWorldPlugin_deserialize_key(
    PRESTypePluginEndpointData endpoint_data,
    HelloWorld **sample,
    RTIBool * drop_sample,
    struct RTICdrStream *cdrStream,
    RTIBool deserialize_encapsulation,
    RTIBool deserialize_key,
    void *endpoint_plugin_qos)
{
    RTIBool result;
    RTIOsapiUtility_unusedParameter(drop_sample);
    /*  Depending on the type and the flags used in rtiddsgen, coverity may detect
    that sample is always null. Since the case is very dependant on
    the IDL/XML and the configuration we keep the check for safety.
    */
    result= PRESTypePlugin_interpretedDeserializeKey(
        endpoint_data,
        /* coverity[check_after_deref] */
        (sample != NULL) ? *sample : NULL,
        cdrStream,
        deserialize_encapsulation,
        deserialize_key,
        endpoint_plugin_qos);
    return result;
}
unsigned int
HelloWorldPlugin_get_serialized_key_max_size(
    PRESTypePluginEndpointData endpoint_data,
    RTIBool include_encapsulation,
    RTIEncapsulationId encapsulation_id,
    unsigned int current_alignment)
{
    unsigned int size;
    RTIBool overflow = RTI_FALSE;
    size = PRESTypePlugin_interpretedGetSerializedKeyMaxSize(
        endpoint_data,&overflow,include_encapsulation,encapsulation_id,current_alignment);
    if (overflow) {
        size = RTI_CDR_MAX_SERIALIZED_SIZE;
    }
    return size;
}
unsigned int
HelloWorldPlugin_get_serialized_key_max_size_for_keyhash(
    PRESTypePluginEndpointData endpoint_data,
    RTIEncapsulationId encapsulation_id,
    unsigned int current_alignment)
{
    unsigned int size;
    RTIBool overflow = RTI_FALSE;
    size = PRESTypePlugin_interpretedGetSerializedKeyMaxSizeForKeyhash(
        endpoint_data,
        &overflow,
        encapsulation_id,
        current_alignment);
    if (overflow) {
        size = RTI_CDR_MAX_SERIALIZED_SIZE;
    }
    return size;
}
struct RTIXCdrInterpreterPrograms * HelloWorldPlugin_get_programs(void)
{
    struct RTIXCdrInterpreterProgramsGenProperty programProperty =
    RTIXCdrInterpreterProgramsGenProperty_INITIALIZER;
    struct RTIXCdrInterpreterPrograms *retPrograms = NULL;
    programProperty.generateWithOnlyKeyFields = RTI_XCDR_FALSE;
    programProperty.generateV1Encapsulation = RTI_XCDR_TRUE;
    programProperty.generateV2Encapsulation = RTI_XCDR_TRUE;
    programProperty.resolveAlias = RTI_XCDR_TRUE;
    programProperty.inlineStruct = RTI_XCDR_TRUE;
    programProperty.optimizeEnum = RTI_XCDR_TRUE;
    programProperty.unboundedSize = RTIXCdrLong_MAX;
    retPrograms =
    DDS_TypeCodeFactory_assert_programs_in_global_list(
        DDS_TypeCodeFactory_get_instance(),
        HelloWorld_get_typecode(),
        &programProperty,
        RTI_XCDR_SER_PROGRAM
        | RTI_XCDR_DESER_PROGRAM
        | RTI_XCDR_GET_MAX_SER_SIZE_PROGRAM
        | RTI_XCDR_GET_SER_SIZE_PROGRAM);
```

```
    return retPrograms;
}
/* ------------------------------------------------------------------------
* Plug-in Installation Methods
* ------------------------------------------------------------------------ */
struct PRESTypePlugin *HelloWorldPlugin_new(void)
{
    struct PRESTypePlugin *plugin = NULL;
    const struct PRESTypePluginVersion PLUGIN_VERSION =
    PRES_TYPE_PLUGIN_VERSION_2_0;
    RTIOsapiHeap_allocateStructure(
        &plugin, struct PRESTypePlugin);
    if (plugin == NULL) {
        return NULL;
    }
    plugin->version = PLUGIN_VERSION;
    /* set up parent's function pointers */
    plugin->onParticipantAttached =
    (PRESTypePluginOnParticipantAttachedCallback)
    HelloWorldPlugin_on_participant_attached;
    plugin->onParticipantDetached =
    (PRESTypePluginOnParticipantDetachedCallback)
    HelloWorldPlugin_on_participant_detached;
    plugin->onEndpointAttached =
    (PRESTypePluginOnEndpointAttachedCallback)
    HelloWorldPlugin_on_endpoint_attached;
    plugin->onEndpointDetached =
    (PRESTypePluginOnEndpointDetachedCallback)
    HelloWorldPlugin_on_endpoint_detached;
    plugin->copySampleFnc =
    (PRESTypePluginCopySampleFunction)
    HelloWorldPlugin_copy_sample;
    plugin->createSampleFnc =
    (PRESTypePluginCreateSampleFunction)
    HelloWorldPlugin_create_sample;
    plugin->destroySampleFnc =
    (PRESTypePluginDestroySampleFunction)
    HelloWorldPlugin_destroy_sample;
    plugin->finalizeOptionalMembersFnc =
    (PRESTypePluginFinalizeOptionalMembersFunction)
    HelloWorldPlugin_finalize_optional_members;
    plugin->serializeFnc =
    (PRESTypePluginSerializeFunction) PRESTypePlugin_interpretedSerialize;
    plugin->deserializeFnc =
    (PRESTypePluginDeserializeFunction) PRESTypePlugin_interpretedDeserializeWithAlloc;
    plugin->getSerializedSampleMaxSizeFnc =
    (PRESTypePluginGetSerializedSampleMaxSizeFunction)
    HelloWorldPlugin_get_serialized_sample_max_size;
    plugin->getSerializedSampleMinSizeFnc =
    (PRESTypePluginGetSerializedSampleMinSizeFunction)
    PRESTypePlugin_interpretedGetSerializedSampleMinSize;
    plugin->getDeserializedSampleMaxSizeFnc = NULL;
    plugin->getSampleFnc =
    (PRESTypePluginGetSampleFunction)
    HelloWorldPlugin_get_sample;
    plugin->returnSampleFnc =
    (PRESTypePluginReturnSampleFunction)
    HelloWorldPlugin_return_sample;
    plugin->getKeyKindFnc =
    (PRESTypePluginGetKeyKindFunction)
    HelloWorldPlugin_get_key_kind;
    /* These functions are only used for keyed types. As this is not a keyed
    type they are all set to NULL
    */
    plugin->serializeKeyFnc = NULL ;
    plugin->deserializeKeyFnc = NULL;
    plugin->getKeyFnc = NULL;
    plugin->returnKeyFnc = NULL;
    plugin->instanceToKeyFnc = NULL;
    plugin->keyToInstanceFnc = NULL;
    plugin->getSerializedKeyMaxSizeFnc = NULL;
    plugin->instanceToKeyHashFnc = NULL;
    plugin->serializedSampleToKeyHashFnc = NULL;
    plugin->serializedKeyToKeyHashFnc = NULL;
    #ifdef NDDS_STANDALONE_TYPE
    plugin->typeCode = NULL;
    #else
    plugin->typeCode =  (struct RTICdrTypeCode *)HelloWorld_get_typecode();
    #endif
    plugin->languageKind = PRES_TYPEPLUGIN_DDS_TYPE;
    /* Serialized buffer */
```

```
    plugin->getBuffer =
    (PRESTypePluginGetBufferFunction)
    HelloWorldPlugin_get_buffer;
    plugin->returnBuffer =
    (PRESTypePluginReturnBufferFunction)
    HelloWorldPlugin_return_buffer;
    plugin->getBufferWithParams = NULL;
    plugin->returnBufferWithParams = NULL;
    plugin->getSerializedSampleSizeFnc =
    (PRESTypePluginGetSerializedSampleSizeFunction)
    PRESTypePlugin_interpretedGetSerializedSampleSize;
    plugin->getWriterLoanedSampleFnc = NULL;
    plugin->returnWriterLoanedSampleFnc = NULL;
    plugin->returnWriterLoanedSampleFromCookieFnc = NULL;
    plugin->validateWriterLoanedSampleFnc = NULL;
    plugin->setWriterLoanedSampleSerializedStateFnc = NULL;
    plugin->endpointTypeName = HelloWorldTYPENAME;
    plugin->isMetpType = RTI_FALSE;
    plugin->isRecursiveType = RTI_FALSE;
    return plugin;
}
void
HelloWorldPlugin_delete(struct PRESTypePlugin *plugin)
{
    RTIOsapiHeap_freeStructure(plugin);
}
#undef RTI_CDR_CURRENT_SUBMODULE
```

5.10 HelloWorldSupport.c

[$(NDDSHOME)/example/C/helloWorldPersistence/HelloWorldSupport.h]

```
/*
WARNING: THIS FILE IS AUTO-GENERATED. DO NOT MODIFY.

This file was generated from HelloWorld.idl
using RTI Code Generator (rtiddsgen) version 4.7.0.
The rtiddsgen tool is part of the RTI Connext DDS distribution.
For more information, type 'rtiddsgen -help' at a command shell
or consult the Code Generator User's Manual.
*/
#ifndef HelloWorldSupport_1436885481_h
#define HelloWorldSupport_1436885481_h
/* Uses */
#include "HelloWorld.h"
#ifndef ndds_c_h
#include "ndds/ndds_c.h"
#endif
#ifdef __cplusplus
extern "C" {
    #endif
    #if (defined(RTI_WIN32) || defined (RTI_WINCE) || defined(RTI_INTIME)) && defined(NDDS_USER_DLL_EXPORT)
    #endif
    /* ========================================================================= */
    #if defined(NDDS_USER_DLL_EXPORT) && defined(RTI_WIN32)
    #undef NDDSUSERDllExport
    #define NDDSUSERDllExport __declspec(dllexport)
    #endif
    #if !defined(RTI_WIN32) && defined(NDDS_USER_SYMBOL_EXPORT)
    #undef NDDSUSERDllExport
    #define NDDSUSERDllExport __attribute__((visibility("default")))
    #endif
    DDS_TYPESUPPORT_C(HelloWorldTypeSupport, HelloWorld);
    DDS_DATAWRITER_WITH_DATA_CONSTRUCTOR_METHODS_C(HelloWorldDataWriter, HelloWorld);
    DDS_DATAREADER_C(HelloWorldDataReader, HelloWorldSeq, HelloWorld);
    #if defined(NDDS_USER_DLL_EXPORT) || defined(NDDS_USER_SYMBOL_EXPORT)
    #undef NDDSUSERDllExport
    #define NDDSUSERDllExport
    #endif
    #ifdef __cplusplus
}
#endif
#endif  /* HelloWorldSupport_1436885481_h */
```

[$(NDDSHOME)/example/C/helloWorldPersistence/HelloWorldSupport.c]

```
/*
WARNING: THIS FILE IS AUTO-GENERATED. DO NOT MODIFY.

This file was generated from HelloWorld.idl
using RTI Code Generator (rtiddsgen) version 4.7.0.
The rtiddsgen tool is part of the RTI Connext DDS distribution.
For more information, type 'rtiddsgen -help' at a command shell
or consult the Code Generator User's Manual.
*/
#include "HelloWorldSupport.h"
#include "HelloWorldPlugin.h"
/* ========================================================================= */
/* ------------------------------------------------------------------- */
/* DDSDataWriter
*/
/* Requires */
#define TTYPENAME   HelloWorldTYPENAME
/* Defines */
#define TDataWriter HelloWorldDataWriter
#define TData       HelloWorld
#define RTI_ENABLE_TDATAWRITER_DATA_CONSTRUCTOR_METHODS
#include "dds_c/generic/dds_c_data_TDataWriter.gen"
#undef RTI_ENABLE_TDATAWRITER_DATA_CONSTRUCTOR_METHODS
#undef TDataWriter
#undef TData
#undef TTYPENAME
/* ------------------------------------------------------------------- */
/* DDSDataReader
*/
/* Requires */
#define TTYPENAME   HelloWorldTYPENAME
/* Defines */
#define TDataReader HelloWorldDataReader
#define TDataSeq    HelloWorldSeq
#define TData       HelloWorld
#define RTI_ENABLE_TDATAREADER_DATA_CONSISTENCY_CHECK_METHOD
#include "dds_c/generic/dds_c_data_TDataReader.gen"
#undef RTI_ENABLE_TDATAREADER_DATA_CONSISTENCY_CHECK_METHOD
#undef TDataReader
#undef TDataSeq
#undef TData
#undef TTYPENAME
/* ------------------------------------------------------------------- */
/* TypeSupport

«IMPLEMENTATION »

Requires:  TTYPENAME,
TPlugin_new
TPlugin_delete
Defines:   TTypeSupport, TData, TDataReader, TDataWriter
*/
/* Requires */
#define TTYPENAME    HelloWorldTYPENAME
#define TPlugin_new  HelloWorldPlugin_new
#define TPlugin_delete  HelloWorldPlugin_delete
/* Defines */
#define TTypeSupport HelloWorldTypeSupport
#define TData        HelloWorld
#define TDataReader  HelloWorldDataReader
#define TDataWriter  HelloWorldDataWriter
#define TGENERATE_SER_CODE
#ifndef NDDS_STANDALONE_TYPE
#define TGENERATE_TYPECODE
#endif
#include "dds_c/generic/dds_c_data_TTypeSupport.gen"
#undef TTypeSupport
#undef TData
#undef TDataReader
#undef TDataWriter
#ifndef NDDS_STANDALONE_TYPE
#undef TGENERATE_TYPECODE
#endif
#undef TGENERATE_SER_CODE
#undef TTYPENAME
#undef TPlugin_new
#undef TPlugin_delete
```

5.11 C++ Example

Persistence Example Using C++.

Modules

- **HelloWorld.idl**
- **HelloWorld.cxx**
- **HelloWorld_publisher.cxx**
- **HelloWorld_subscriber.cxx**
- **HelloWorldPlugin.cxx**
- **HelloWorldSupport.cxx**

5.11.1 Detailed Description

Persistence Example Using C++.

See README.txt for details.

[$(NDDSHOME)/example/CPP/helloWorldPersistence/README.txt]

5.12 HelloWorld.idl

[$(NDDSHOME)/example/CPP/helloWorldPersistence/HelloWorld.idl]

```
struct HelloWorld {
    long data;
};
```

5.13 HelloWorld.cxx

[$(NDDSHOME)/example/CPP/helloWorldPersistence/HelloWorld.h]

```
/*
WARNING: THIS FILE IS AUTO-GENERATED. DO NOT MODIFY.

This file was generated from HelloWorld.idl
using RTI Code Generator (rtiddsgen) version 4.7.0.
The rtiddsgen tool is part of the RTI Connext DDS distribution.
For more information, type 'rtiddsgen -help' at a command shell
or consult the Code Generator User's Manual.
*/
#ifndef HelloWorld_1436885481_h
#define HelloWorld_1436885481_h
#ifndef NDDS_STANDALONE_TYPE
#ifndef ndds_c_h
#include "ndds/ndds_c.h"
#endif
#include "cdr/cdr_typeCode.h"
#else
#include "ndds_standalone_type.h"
#endif
#ifdef __cplusplus
extern "C" {
    #endif
```

```
    extern const char *HelloWorldTYPENAME;
    typedef struct HelloWorld
    {
        DDS_Long data;
    } HelloWorld ;
    #if defined(NDDS_USER_DLL_EXPORT) && defined(RTI_WIN32)
    #undef NDDSUSERDllExport
    #define NDDSUSERDllExport __declspec(dllexport)
    #endif
    #if !defined(RTI_WIN32) && defined(NDDS_USER_SYMBOL_EXPORT)
    #undef NDDSUSERDllExport
    #define NDDSUSERDllExport __attribute__((visibility("default")))
    #endif
    #ifndef NDDS_STANDALONE_TYPE
    NDDSUSERDllExport DDS_TypeCode * HelloWorld_get_typecode(void); /* Type code */
    NDDSUSERDllExport RTIXCdrTypePlugin *HelloWorld_get_type_plugin_info(void);
    NDDSUSERDllExport RTIXCdrSampleAccessInfo *HelloWorld_get_sample_access_info(void);
    #endif
    DDS_SEQUENCE(HelloWorldSeq, HelloWorld);
    NDDSUSERDllExport
    RTIBool HelloWorld_initialize(
        HelloWorld* self);
    NDDSUSERDllExport
    RTIBool HelloWorld_initialize_ex(
        HelloWorld* self,RTIBool allocatePointers,RTIBool allocateMemory);
    NDDSUSERDllExport
    RTIBool HelloWorld_initialize_w_params(
        HelloWorld* self,
        const struct DDS_TypeAllocationParams_t * allocParams);
    NDDSUSERDllExport
    RTIBool HelloWorld_finalize_w_return(
        HelloWorld* self);
    NDDSUSERDllExport
    void HelloWorld_finalize(
        HelloWorld* self);
    NDDSUSERDllExport
    void HelloWorld_finalize_ex(
        HelloWorld* self,RTIBool deletePointers);
    NDDSUSERDllExport
    void HelloWorld_finalize_w_params(
        HelloWorld* self,
        const struct DDS_TypeDeallocationParams_t * deallocParams);
    NDDSUSERDllExport
    void HelloWorld_finalize_optional_members(
        HelloWorld* self, RTIBool deletePointers);
    NDDSUSERDllExport
    RTIBool HelloWorld_copy(
        HelloWorld* dst,
        const HelloWorld* src);
    #if defined(NDDS_USER_DLL_EXPORT) || defined(NDDS_USER_SYMBOL_EXPORT)
    #undef NDDSUSERDllExport
    #define NDDSUSERDllExport
    #endif
    #ifdef __cplusplus
}
#endif
#endif /* HelloWorld */
```

[$(NDDSHOME)/example/CPP/helloWorldPersistence/HelloWorld.cxx]

```
/*
WARNING: THIS FILE IS AUTO-GENERATED. DO NOT MODIFY.

This file was generated from HelloWorld.idl
using RTI Code Generator (rtiddsgen) version 4.7.0.
The rtiddsgen tool is part of the RTI Connext DDS distribution.
For more information, type 'rtiddsgen -help' at a command shell
or consult the Code Generator User's Manual.
*/
#ifndef NDDS_STANDALONE_TYPE
#ifndef ndds_cpp_h
#include "ndds/ndds_cpp.h"
#endif
#ifndef dds_c_log_impl_h
#include "dds_c/dds_c_log_impl.h"
#endif
#ifndef dds_c_log_infrastructure_h
#include "dds_c/dds_c_infrastructure_impl.h"
#endif
#ifndef cdr_type_h
#include "cdr/cdr_type.h"
```

```
#endif
#include "osapi/osapi_atomic.h"
#else
#include "ndds_standalone_type.h"
#endif
#include "HelloWorld.h"
#ifndef NDDS_STANDALONE_TYPE
#include "HelloWorldPlugin.h"
#endif
#include <new>
/* ========================================================================= */
const char *HelloWorldTYPENAME = "HelloWorld";
#ifndef NDDS_STANDALONE_TYPE
DDS_TypeCode * HelloWorld_get_typecode(void)
{
    static RTI_ATOMIC(RTIBool) is_initialized;
    static DDS_TypeCode_Member HelloWorld_g_tc_members[1]=
    {
        {
            (char *)"data",/* Member name */
            {
                0,/* Representation ID */
                DDS_BOOLEAN_FALSE,/* Is a pointer? */
                -1, /* Bitfield bits */
                NULL/* Member type code is assigned later */
            },
            0, /* Ignored */
            0, /* Ignored */
            0, /* Ignored */
            NULL, /* Ignored */
            RTI_CDR_REQUIRED_MEMBER, /* Is a key? */
            DDS_PUBLIC_MEMBER,/* Member visibility */
            RTICdrTypeCodeAnnotations_INITIALIZER
        }
    };
    static DDS_TypeCode HelloWorld_g_tc =
    {{
            DDS_TK_STRUCT, /* Kind */
            DDS_BOOLEAN_FALSE, /* Ignored */
            -1, /*Ignored*/
            (char *)"HelloWorld", /* Name */
            NULL, /* Ignored */
            0, /* Ignored */
            0, /* Ignored */
            NULL, /* Ignored */
            1, /* Number of members */
            HelloWorld_g_tc_members, /* Members */
            DDS_VM_NONE, /* Ignored */
            RTICdrTypeCodeAnnotations_INITIALIZER,
            DDS_BOOLEAN_TRUE, /* _isCopyable */
            NULL, /* _sampleAccessInfo: assigned later */
            NULL /* _typePlugin: assigned later */
        }}; /* Type code for HelloWorld*/
    if (RTIOsapiAtomic_load32(&is_initialized, RTI_OSAPI_ATOMIC_MEMORY_ORDER_ACQUIRE)) {
        return &HelloWorld_g_tc;
    }
    HelloWorld_g_tc._data._annotations._allowedDataRepresentationMask = 5;
    HelloWorld_g_tc_members[0]._representation._typeCode =  (RTICdrTypeCode *)&DDS_g_tc_long_w_new;
    /* Initialize the values for member annotations. */
    HelloWorld_g_tc_members[0]._annotations._defaultValue._d = RTI_XCDR_TK_LONG;
    HelloWorld_g_tc_members[0]._annotations._defaultValue._u.long_value = 0;
    HelloWorld_g_tc_members[0]._annotations._minValue._d = RTI_XCDR_TK_LONG;
    HelloWorld_g_tc_members[0]._annotations._minValue._u.long_value = RTIXCdrLong_MIN;
    HelloWorld_g_tc_members[0]._annotations._maxValue._d = RTI_XCDR_TK_LONG;
    HelloWorld_g_tc_members[0]._annotations._maxValue._u.long_value = RTIXCdrLong_MAX;
    HelloWorld_g_tc._data._sampleAccessInfo =
    HelloWorld_get_sample_access_info();
    HelloWorld_g_tc._data._typePlugin =
    HelloWorld_get_type_plugin_info();
    RTIOsapiAtomic_store32(
        &is_initialized,
        RTI_TRUE,
        RTI_OSAPI_ATOMIC_MEMORY_ORDER_RELEASE);
    return &HelloWorld_g_tc;
}
#define TSeq HelloWorldSeq
#define T HelloWorld
#include "dds_cpp/generic/dds_cpp_data_TInterpreterSupport.gen"
#undef T
#undef TSeq
RTIXCdrSampleAccessInfo *HelloWorld_get_sample_seq_access_info()
```

```
{
    static RTIXCdrSampleAccessInfo HelloWorld_g_seqSampleAccessInfo = {
        RTI_XCDR_TYPE_BINDING_CPP, \
        {sizeof(HelloWorldSeq),0,0,0}, \
        RTI_XCDR_FALSE, \
        DDS_Sequence_get_member_value_pointer, \
        HelloWorldSeq_set_member_element_count, \
        NULL, \
        NULL, \
        NULL \
    };
    return &HelloWorld_g_seqSampleAccessInfo;
}
RTIXCdrSampleAccessInfo *HelloWorld_get_sample_access_info()
{
    static RTI_ATOMIC(RTIBool) is_initialized;
    HelloWorld *sample;
    static RTIXCdrMemberAccessInfo HelloWorld_g_memberAccessInfos[1] =
    {RTIXCdrMemberAccessInfo_INITIALIZER};
    static RTIXCdrSampleAccessInfo HelloWorld_g_sampleAccessInfo =
    RTIXCdrSampleAccessInfo_INITIALIZER;
    if (RTIOsapiAtomic_load32(
        &is_initialized,
        RTI_OSAPI_ATOMIC_MEMORY_ORDER_ACQUIRE)) {
        return (RTIXCdrSampleAccessInfo*) &HelloWorld_g_sampleAccessInfo;
    }
    RTIXCdrHeap_allocateStruct(
        &sample,
        HelloWorld);
    if (sample == NULL) {
        return NULL;
    }
    HelloWorld_g_memberAccessInfos[0].bindingMemberValueOffset[0] =
    (RTIXCdrUnsignedLong) ((char *)&sample->data - (char *)sample);
    HelloWorld_g_sampleAccessInfo.memberAccessInfos =
    HelloWorld_g_memberAccessInfos;
    {
        size_t candidateTypeSize = sizeof(HelloWorld);
        if (candidateTypeSize > RTIXCdrLong_MAX) {
            HelloWorld_g_sampleAccessInfo.typeSize[0] =
            RTIXCdrLong_MAX;
        } else {
            HelloWorld_g_sampleAccessInfo.typeSize[0] =
            (RTIXCdrUnsignedLong) candidateTypeSize;
        }
    }
    HelloWorld_g_sampleAccessInfo.useGetMemberValueOnlyWithRef =
    RTI_XCDR_TRUE;
    HelloWorld_g_sampleAccessInfo.getMemberValuePointerFcn =
    HelloWorld_get_member_value_pointer;
    HelloWorld_g_sampleAccessInfo.languageBinding =
    RTI_XCDR_TYPE_BINDING_CPP ;
    RTIXCdrHeap_freeStruct(sample);
    RTIOsapiAtomic_store32(
        &is_initialized,
        RTI_TRUE,
        RTI_OSAPI_ATOMIC_MEMORY_ORDER_RELEASE);
    return (RTIXCdrSampleAccessInfo*) &HelloWorld_g_sampleAccessInfo;
}
RTIXCdrTypePlugin *HelloWorld_get_type_plugin_info()
{
    static RTIXCdrTypePlugin HelloWorld_g_typePlugin =
    {
        NULL, /* serialize */
        NULL, /* serialize_key */
        NULL, /* deserialize_sample */
        NULL, /* deserialize_key_sample */
        NULL, /* skip */
        NULL, /* get_serialized_sample_size */
        NULL, /* get_serialized_sample_max_size_ex */
        NULL, /* get_serialized_key_max_size_ex */
        NULL, /* get_serialized_sample_min_size */
        NULL, /* serialized_sample_to_key */
        (RTIXCdrTypePluginInitializeSampleFunction)
        HelloWorld_initialize_ex,
        NULL,
        (RTIXCdrTypePluginFinalizeSampleFunction)
        HelloWorld_finalize_w_return,
        NULL,
        NULL
    };
```

```
    return &HelloWorld_g_typePlugin;
}
#endif
RTIBool HelloWorld_initialize(
    HelloWorld* sample)
{
    return HelloWorld_initialize_ex(
        sample,
        RTI_TRUE,
        RTI_TRUE);
}
RTIBool HelloWorld_initialize_w_params(
    HelloWorld *sample,
    const struct DDS_TypeAllocationParams_t *allocParams)
{
    if (sample == NULL) {
        return RTI_FALSE;
    }
    if (allocParams == NULL) {
        return RTI_FALSE;
    }
    sample->data = 0;
    return RTI_TRUE;
}
RTIBool HelloWorld_initialize_ex(
    HelloWorld *sample,
    RTIBool allocatePointers,
    RTIBool allocateMemory)
{
    struct DDS_TypeAllocationParams_t allocParams =
    DDS_TYPE_ALLOCATION_PARAMS_DEFAULT;
    allocParams.allocate_pointers =  (DDS_Boolean)allocatePointers;
    allocParams.allocate_memory = (DDS_Boolean)allocateMemory;
    return HelloWorld_initialize_w_params(
        sample,
        &allocParams);
}
RTIBool HelloWorld_finalize_w_return(
    HelloWorld* sample)
{
    HelloWorld_finalize_ex(sample, RTI_TRUE);
    return RTI_TRUE;
}
void HelloWorld_finalize(
    HelloWorld* sample)
{
    HelloWorld_finalize_ex(
        sample,
        RTI_TRUE);
}
void HelloWorld_finalize_ex(
    HelloWorld *sample,
    RTIBool deletePointers)
{
    struct DDS_TypeDeallocationParams_t deallocParams =
    DDS_TYPE_DEALLOCATION_PARAMS_DEFAULT;
    if (sample==NULL) {
        return;
    }
    deallocParams.delete_pointers = (DDS_Boolean)deletePointers;
    HelloWorld_finalize_w_params(
        sample,
        &deallocParams);
}
void HelloWorld_finalize_w_params(
    HelloWorld *sample,
    const struct DDS_TypeDeallocationParams_t *deallocParams)
{
    if (sample==NULL) {
        return;
    }
    if (deallocParams == NULL) {
        return;
    }
}
void HelloWorld_finalize_optional_members(
    HelloWorld* sample, RTIBool deletePointers)
{
    struct DDS_TypeDeallocationParams_t deallocParamsTmp =
    DDS_TYPE_DEALLOCATION_PARAMS_DEFAULT;
    struct DDS_TypeDeallocationParams_t * deallocParams =
```

```
    &deallocParamsTmp;
    if (sample==NULL) {
        return;
    }
    RTIOsapiUtility_unusedParameter(deallocParams);
    deallocParamsTmp.delete_pointers = (DDS_Boolean)deletePointers;
    deallocParamsTmp.delete_optional_members = DDS_BOOLEAN_TRUE;
}
RTIBool HelloWorld_copy(
    HelloWorld* dst,
    const HelloWorld* src)
{
    try {
        if (dst == NULL || src == NULL) {
            return RTI_FALSE;
        }
        if (!RTICdrType_copyLong (
            &dst->data,
            &src->data)) {
            return RTI_FALSE;
        }
        return RTI_TRUE;
    } catch (const std::bad_alloc&) {
        return RTI_FALSE;
    }
}
#define T HelloWorld
#define TSeq HelloWorldSeq
#define T_initialize_w_params HelloWorld_initialize_w_params
#define T_finalize_w_params   HelloWorld_finalize_w_params
#define T_copy       HelloWorld_copy
#ifndef NDDS_STANDALONE_TYPE
#include "dds_c/generic/dds_c_sequence_TSeq.gen"
#include "dds_cpp/generic/dds_cpp_sequence_TSeq.gen"
#else
#include "dds_c_sequence_TSeq.gen"
#include "dds_cpp_sequence_TSeq.gen"
#endif
#undef T_copy
#undef T_finalize_w_params
#undef T_initialize_w_params
#undef TSeq
#undef T
#ifndef NDDS_STANDALONE_TYPE
namespace rti {
    namespace xcdr {
        const RTIXCdrTypeCode * type_code< HelloWorld>::get()
        {
            return (const RTIXCdrTypeCode *) HelloWorld_get_typecode();
        }
    }
}
#endif
```

5.14 HelloWorld_publisher.cxx

[$(NDDSHOME)/example/CPP/helloWorldPersistence/HelloWorld_publisher.cxx]

```
/* HelloWorld_publisher.cxx

   A publication of data of type HelloWorld

   This file is derived from code automatically generated by the rtiddsgen
   command:

   rtiddsgen -language C++ -example <arch> HelloWorld.idl

modification history
------------ -------
*/
#include <stdio.h>
#include <stdlib.h>
#include "ndds/ndds_cpp.h"
#include "HelloWorld.h"
#include "HelloWorldSupport.h"
/* Delete all entities */
```

```
static int publisher_shutdown(
    DDSDomainParticipant *participant)
{
    DDS_ReturnCode_t retcode;
    int status = 0;
    if (participant != NULL) {
        retcode = participant->delete_contained_entities();
        if (retcode != DDS_RETCODE_OK) {
            printf("delete_contained_entities error %d\n", retcode);
            status = -1;
        }
        retcode = DDSTheParticipantFactory->delete_participant(participant);
        if (retcode != DDS_RETCODE_OK) {
            printf("delete_participant error %d\n", retcode);
            status = -1;
        }
    }
    /* RTI Connext provides finalize_instance() method for
       people who want to release memory used by the participant factory
       singleton. Uncomment the following block of code for clean destruction of
       the participant factory singleton. */
/*
    retcode = DDSDomainParticipantFactory::finalize_instance();
    if (retcode != DDS_RETCODE_OK) {
        printf("finalize_instance error %d\n", retcode);
        status = -1;
    }
*/
    return status;
}
extern "C" int publisher_main(int domainId, int sample_count)
{
    DDSDomainParticipant *participant = NULL;
    DDSPublisher *publisher = NULL;
    DDSTopic *topic = NULL;
    DDSDataWriter *writer = NULL;
    HelloWorldDataWriter * HelloWorld_writer = NULL;
    HelloWorld *instance = NULL;
    DDS_ReturnCode_t retcode;
    DDS_InstanceHandle_t instance_handle = DDS_HANDLE_NIL;
    const char *type_name = NULL;
    int count = 0;
    struct DDS_Duration_t send_period = {4,0};

    /* To customize participant QoS, use
       DDS_DomainParticipantFactory_get_default_participant_qos() */
    participant = DDSDomainParticipantFactory::get_instance()->create_participant(
        domainId, DDS_PARTICIPANT_QOS_DEFAULT,
        NULL /* listener */, DDS_STATUS_MASK_NONE);
    if (participant == NULL) {
        printf("create_participant error\n");
        publisher_shutdown(participant);
        return -1;
    }
    /* To customize publisher QoS, use
       participant->get_default_publisher_qos() */
    publisher = participant->create_publisher(
        DDS_PUBLISHER_QOS_DEFAULT, NULL /* listener */, DDS_STATUS_MASK_NONE);
    if (publisher == NULL) {
        printf("create_publisher error\n");
        publisher_shutdown(participant);
        return -1;
    }
    /* Register type before creating topic */
    type_name = HelloWorldTypeSupport::get_type_name();
    retcode = HelloWorldTypeSupport::register_type(
        participant, type_name);
    if (retcode != DDS_RETCODE_OK) {
        printf("register_type error %d\n", retcode);
        publisher_shutdown(participant);
        return -1;
    }
    /* To customize topic QoS, use
       participant->get_default_topic_qos() */
    topic = participant->create_topic(
        "Example HelloWorld",
        type_name, DDS_TOPIC_QOS_DEFAULT, NULL /* listener */,
        DDS_STATUS_MASK_NONE);
    if (topic == NULL) {
        printf("create_topic error\n");
        publisher_shutdown(participant);
```

```
        return -1;
    }
    /* Create the datawriter */
    writer = publisher->create_datawriter(
        topic, DDS_DATAWRITER_QOS_DEFAULT, NULL /* listener */,
        DDS_STATUS_MASK_NONE);
    if (writer == NULL) {
        printf("create_datawriter error\n");
        publisher_shutdown(participant);
        return -1;
    }
    HelloWorld_writer = HelloWorldDataWriter::narrow(writer);
    if (HelloWorld_writer == NULL) {
        printf("DataWriter narrow error\n");
        publisher_shutdown(participant);
        return -1;
    }
    /* Create data sample for writing */
    instance = HelloWorldTypeSupport::create_data();
    if (instance == NULL) {
        printf("HelloWorldTypeSupport::create_data error\n");
        publisher_shutdown(participant);
        return -1;
    }
    /* For data type that has key, if the same instance is going to be
       written multiple times, initialize the key here
       and register the keyed instance prior to writing */
/*
    instance_handle = HelloWorld_writer->register_instance(*instance);
*/
    /* Main loop */
    for (count=0; (sample_count == 0) || (count < sample_count); ++count) {
        printf("Writing HelloWorld, count %d\n", count);
        fflush(stdout);
        /* Modify the data to be sent here */
        instance->data = count;
        retcode = HelloWorld_writer->write(*instance, instance_handle);
        if (retcode != DDS_RETCODE_OK) {
            printf("write error %d\n", retcode);
        }
        NDDSUtility::sleep(send_period);
    }
/*
    retcode = HelloWorld_writer->unregister_instance(
        *instance, instance_handle);
    if (retcode != DDS_RETCODE_OK) {
        printf("unregister instance error %d\n", retcode);
    }
*/
    /* Delete data sample */
    retcode = HelloWorldTypeSupport::delete_data(instance);
    if (retcode != DDS_RETCODE_OK) {
        printf("HelloWorldTypeSupport::delete_data error %d\n", retcode);
    }

    /* Delete all entities */
    return publisher_shutdown(participant);
}
#if defined(RTI_WINCE)
int wmain(int argc, wchar_t** argv)
{
    int domainId = 0;
    int sample_count = 0; /* infinite loop */

    if (argc >= 2) {
        domainId = _wtoi(argv[1]);
    }
    if (argc >= 3) {
        sample_count = _wtoi(argv[2]);
    }
     /* Uncomment this to turn on additional logging
    NDDSConfigLogger::get_instance()->
        set_verbosity_by_category(NDDS_CONFIG_LOG_CATEGORY_API,
                                  NDDS_CONFIG_LOG_VERBOSITY_STATUS_ALL);
    */

    return publisher_main(domainId, sample_count);
}

#elif !(defined(RTI_VXWORKS) && !defined(__RTP__)) && !defined(RTI_PSOS)
int main(int argc, char *argv[])
```

```
{
    int domainId = 0;
    int sample_count = 0; /* infinite loop */
    if (argc >= 2) {
        domainId = atoi(argv[1]);
    }
    if (argc >= 3) {
        sample_count = atoi(argv[2]);
    }
    /* Uncomment this to turn on additional logging
    NDDSConfigLogger::get_instance()->
        set_verbosity_by_category(NDDS_CONFIG_LOG_CATEGORY_API,
                                  NDDS_CONFIG_LOG_VERBOSITY_STATUS_ALL);
    */

    return publisher_main(domainId, sample_count);
}
#endif
```

5.15 HelloWorld_subscriber.cxx

[$(NDDSHOME)/example/CPP/helloWorldPersistence/HelloWorld_subscriber.cxx]

```
/* HelloWorld_subscriber.cxx

   A subscription example

   This file is derived from code automatically generated by the rtiddsgen
   command:

   rtiddsgen -language C++ -example <arch> HelloWorld.idl

modification history
------------ -------
*/
#include <stdio.h>
#include <stdlib.h>
#include "ndds/ndds_cpp.h"
#include "HelloWorld.h"
#include "HelloWorldSupport.h"
class HelloWorldListener : public DDSDataReaderListener {
  public:
    virtual void on_requested_deadline_missed(
        DDSDataReader* /*reader*/,
        const DDS_RequestedDeadlineMissedStatus& /*status*/) {}

    virtual void on_requested_incompatible_qos(
        DDSDataReader* /*reader*/,
        const DDS_RequestedIncompatibleQosStatus& /*status*/) {}

    virtual void on_sample_rejected(
        DDSDataReader* /*reader*/,
        const DDS_SampleRejectedStatus& /*status*/) {}
    virtual void on_liveliness_changed(
        DDSDataReader* /*reader*/,
        const DDS_LivelinessChangedStatus& /*status*/) {}
    virtual void on_sample_lost(
        DDSDataReader* /*reader*/,
        const DDS_SampleLostStatus& /*status*/) {}
    virtual void on_subscription_matched(
        DDSDataReader* /*reader*/,
        const DDS_SubscriptionMatchedStatus& /*status*/) {}
    virtual void on_data_available(DDSDataReader* reader);
};
void HelloWorldListener::on_data_available(DDSDataReader* reader)
{
    HelloWorldDataReader *HelloWorld_reader = NULL;
    HelloWorldSeq data_seq;
    DDS_SampleInfoSeq info_seq;
    DDS_ReturnCode_t retcode;
    int i;
    HelloWorld_reader = HelloWorldDataReader::narrow(reader);
    if (HelloWorld_reader == NULL) {
        printf("DataReader narrow error\n");
        return;
    }
```

```
    retcode = HelloWorld_reader->take(
        data_seq, info_seq, DDS_LENGTH_UNLIMITED,
        DDS_ANY_SAMPLE_STATE, DDS_ANY_VIEW_STATE, DDS_ANY_INSTANCE_STATE);
    if (retcode == DDS_RETCODE_NO_DATA) {
        return;
    } else if (retcode != DDS_RETCODE_OK) {
        printf("take error %d\n", retcode);
        return;
    }
    for (i = 0; i < data_seq.length(); ++i) {
        if (info_seq[i].valid_data) {
            HelloWorldTypeSupport::print_data(&data_seq[i]);
        }
    }
    retcode = HelloWorld_reader->return_loan(data_seq, info_seq);
    if (retcode != DDS_RETCODE_OK) {
        printf("return loan error %d\n", retcode);
    }
}
/* Delete all entities */
static int subscriber_shutdown(
    DDSDomainParticipant *participant)
{
    DDS_ReturnCode_t retcode;
    int status = 0;
    if (participant != NULL) {
        retcode = participant->delete_contained_entities();
        if (retcode != DDS_RETCODE_OK) {
            printf("delete_contained_entities error %d\n", retcode);
            status = -1;
        }
        retcode = DDSTheParticipantFactory->delete_participant(participant);
        if (retcode != DDS_RETCODE_OK) {
            printf("delete_participant error %d\n", retcode);
            status = -1;
        }
    }
    /* RTI Connext provides finalize_instance() method for
       people who want to release memory used by the participant factory
       singleton. Uncomment the following block of code for clean destruction of
       the participant factory singleton. */
/*
    retcode = DDSDomainParticipantFactory::finalize_instance();
    if (retcode != DDS_RETCODE_OK) {
        printf("finalize_instance error %d\n", retcode);
        status = -1;
    }
*/
    return status;
}
extern "C" int subscriber_main(int domainId, int sample_count)
{
    DDSDomainParticipant *participant = NULL;
    DDSSubscriber *subscriber = NULL;
    DDSTopic *topic = NULL;
    HelloWorldListener *reader_listener = NULL;
    DDSDataReader *reader = NULL;
    DDS_ReturnCode_t retcode;
    const char *type_name = NULL;
    int count = 0;
    struct DDS_Duration_t receive_period = {4,0};
    int status = 0;
    /* To customize participant QoS, use
       DDS_DomainParticipantFactory_get_default_participant_qos() */
    participant = DDSDomainParticipantFactory::get_instance()->create_participant(
        domainId, DDS_PARTICIPANT_QOS_DEFAULT,
        NULL /* listener */, DDS_STATUS_MASK_NONE);
    if (participant == NULL) {
        printf("create_participant error\n");
        subscriber_shutdown(participant);
        return -1;
    }
    /* To customize subscriber QoS, use
       participant->get_default_subscriber_qos() */
    subscriber = participant->create_subscriber(
        DDS_SUBSCRIBER_QOS_DEFAULT, NULL /* listener */, DDS_STATUS_MASK_NONE);
    if (subscriber == NULL) {
        printf("create_subscriber error\n");
        subscriber_shutdown(participant);
        return -1;
    }
```

```
    /* Register type before creating topic */
    type_name = HelloWorldTypeSupport::get_type_name();
    retcode = HelloWorldTypeSupport::register_type(
        participant, type_name);
    if (retcode != DDS_RETCODE_OK) {
        printf("register_type error %d\n", retcode);
        subscriber_shutdown(participant);
        return -1;
    }
    /* To customize topic QoS, use
       participant->get_default_topic_qos() */
    topic = participant->create_topic(
        "Example HelloWorld",
        type_name, DDS_TOPIC_QOS_DEFAULT, NULL /* listener */,
        DDS_STATUS_MASK_NONE);
    if (topic == NULL) {
        printf("create_topic error\n");
        subscriber_shutdown(participant);
        return -1;
    }
    /* Create data datareader listener */
    reader_listener = new HelloWorldListener();
    if (reader_listener ==  NULL) {
        printf("listener instantiation error\n");
        subscriber_shutdown(participant);
        return -1;
    }
    /* Create the datareader */
    reader = subscriber->create_datareader(
        topic, DDS_DATAREADER_QOS_DEFAULT, reader_listener,
        DDS_STATUS_MASK_ALL);
    if (reader == NULL) {
        printf("create_datareader error\n");
        subscriber_shutdown(participant);
        delete reader_listener;
        return -1;
    }
    /* Main loop */
    for (count=0; (sample_count == 0) || (count < sample_count); ++count) {
        printf("HelloWorld subscriber sleeping for %d sec...\n",
               receive_period.sec);
        fflush(stdout);
        NDDSUtility::sleep(receive_period);
    }
    /* Delete all entities */
    status = subscriber_shutdown(participant);
    delete reader_listener;
    return status;
}
#if defined(RTI_WINCE)
int wmain(int argc, wchar_t** argv)
{
    int domainId = 0;
    int sample_count = 0; /* infinite loop */

    if (argc >= 2) {
        domainId = _wtoi(argv[1]);
    }
    if (argc >= 3) {
        sample_count = _wtoi(argv[2]);
    }

    /* Uncomment this to turn on additional logging
    NDDSConfigLogger::get_instance()->
        set_verbosity_by_category(NDDS_CONFIG_LOG_CATEGORY_API,
                                  NDDS_CONFIG_LOG_VERBOSITY_STATUS_ALL);
    */

    return subscriber_main(domainId, sample_count);
}
#elif !(defined(RTI_VXWORKS) && !defined(__RTP__)) && !defined(RTI_PSOS)
int main(int argc, char *argv[])
{
    int domainId = 0;
    int sample_count = 0; /* infinite loop */
    if (argc >= 2) {
        domainId = atoi(argv[1]);
    }
    if (argc >= 3) {
        sample_count = atoi(argv[2]);
    }
```

```
    /* Uncomment this to turn on additional logging
    NDDSConfigLogger::get_instance()->
        set_verbosity_by_category(NDDS_CONFIG_LOG_CATEGORY_API,
                                  NDDS_CONFIG_LOG_VERBOSITY_STATUS_ALL);
    */

    return subscriber_main(domainId, sample_count);
}
#endif
```

5.16 HelloWorldPlugin.cxx

[$(NDDSHOME)/example/CPP/helloWorldPersistence/HelloWorldPlugin.h]

```
/*
WARNING: THIS FILE IS AUTO-GENERATED. DO NOT MODIFY.

This file was generated from HelloWorld.idl
using RTI Code Generator (rtiddsgen) version 4.7.0.
The rtiddsgen tool is part of the RTI Connext DDS distribution.
For more information, type 'rtiddsgen -help' at a command shell
or consult the Code Generator User's Manual.
*/
#ifndef HelloWorldPlugin_1436885481_h
#define HelloWorldPlugin_1436885481_h
#include "HelloWorld.h"
struct RTICdrStream;
#ifndef pres_typePlugin_h
#include "pres/pres_typePlugin.h"
#endif
#if defined(NDDS_USER_DLL_EXPORT) && defined(RTI_WIN32)
#undef NDDSUSERDllExport
#define NDDSUSERDllExport __declspec(dllexport)
#endif
#if !defined(RTI_WIN32) && defined(NDDS_USER_SYMBOL_EXPORT)
#undef NDDSUSERDllExport
#define NDDSUSERDllExport __attribute__((visibility("default")))
#endif
#ifdef __cplusplus
extern "C" {
    #endif
    #define HelloWorldPlugin_get_sample PRESTypePluginDefaultEndpointData_getSample
    #define HelloWorldPlugin_get_buffer PRESTypePluginDefaultEndpointData_getBuffer
    #define HelloWorldPlugin_return_buffer PRESTypePluginDefaultEndpointData_returnBuffer
    #define HelloWorldPlugin_create_sample PRESTypePluginDefaultEndpointData_createSample
    #define HelloWorldPlugin_destroy_sample PRESTypePluginDefaultEndpointData_deleteSample
    /* --------------------------------------------------------------------------------------
    Support functions:
    * -------------------------------------------------------------------------------------- */
    NDDSUSERDllExport extern HelloWorld*
    HelloWorldPluginSupport_create_data_w_params(
        const struct DDS_TypeAllocationParams_t * alloc_params);
    NDDSUSERDllExport extern HelloWorld*
    HelloWorldPluginSupport_create_data_ex(RTIBool allocate_pointers);
    NDDSUSERDllExport extern HelloWorld*
    HelloWorldPluginSupport_create_data(void);
    NDDSUSERDllExport extern RTIBool
    HelloWorldPluginSupport_copy_data(
        HelloWorld *out,
        const HelloWorld *in);
    NDDSUSERDllExport extern void
    HelloWorldPluginSupport_destroy_data_w_params(
        HelloWorld *sample,
        const struct DDS_TypeDeallocationParams_t * dealloc_params);
    NDDSUSERDllExport extern void
    HelloWorldPluginSupport_destroy_data_ex(
        HelloWorld *sample,RTIBool deallocate_pointers);
    NDDSUSERDllExport extern void
    HelloWorldPluginSupport_destroy_data(
        HelloWorld *sample);
    NDDSUSERDllExport extern void
    HelloWorldPluginSupport_print_data(
        const HelloWorld *sample,
        const char *desc,
        unsigned int indent);
    /* --------------------------------------------------------------------------
```

```
    Callback functions:
    * ------------------------------------------------------------------------- */
    NDDSUSERDllExport extern PRESTypePluginParticipantData
    HelloWorldPlugin_on_participant_attached(
        void *registration_data,
        const struct PRESTypePluginParticipantInfo *participant_info,
        RTIBool top_level_registration,
        void *container_plugin_context,
        RTICdrTypeCode *typeCode);
    NDDSUSERDllExport extern void
    HelloWorldPlugin_on_participant_detached(
        PRESTypePluginParticipantData participant_data);
    NDDSUSERDllExport extern PRESTypePluginEndpointData
    HelloWorldPlugin_on_endpoint_attached(
        PRESTypePluginParticipantData participant_data,
        const struct PRESTypePluginEndpointInfo *endpoint_info,
        RTIBool top_level_registration,
        void *container_plugin_context);
    NDDSUSERDllExport extern void
    HelloWorldPlugin_on_endpoint_detached(
        PRESTypePluginEndpointData endpoint_data);
    NDDSUSERDllExport extern void
    HelloWorldPlugin_return_sample(
        PRESTypePluginEndpointData endpoint_data,
        HelloWorld *sample,
        void *handle);
    NDDSUSERDllExport extern RTIBool
    HelloWorldPlugin_copy_sample(
        PRESTypePluginEndpointData endpoint_data,
        HelloWorld *out,
        const HelloWorld *in);
    /* ----------------------------------------------------------------------------
    (De)Serialize functions:
    * ------------------------------------------------------------------------- */
    NDDSUSERDllExport extern RTIBool
    HelloWorldPlugin_serialize_to_cdr_buffer(
        char * buffer,
        unsigned int * length,
        const HelloWorld *sample);
    NDDSUSERDllExport extern RTIBool
    HelloWorldPlugin_serialize_to_cdr_buffer_ex(
        char *buffer,
        unsigned int *length,
        const HelloWorld *sample,
        DDS_DataRepresentationId_t representation);
    NDDSUSERDllExport extern RTIBool
    HelloWorldPlugin_deserialize_from_cdr_buffer(
        HelloWorld *sample,
        const char * buffer,
        unsigned int length);
    #if !defined (NDDS_STANDALONE_TYPE)
    NDDSUSERDllExport extern DDS_ReturnCode_t
    HelloWorldPlugin_data_to_string(
        const HelloWorld *sample,
        char *str,
        DDS_UnsignedLong *str_size,
        const struct DDS_PrintFormatProperty *property);
    #endif
    NDDSUSERDllExport extern unsigned int
    HelloWorldPlugin_get_serialized_sample_max_size(
        PRESTypePluginEndpointData endpoint_data,
        RTIBool include_encapsulation,
        RTIEncapsulationId encapsulation_id,
        unsigned int current_alignment);
    /* ----------------------------------------------------------------------------
    Key Management functions:
    * ------------------------------------------------------------------------- */
    NDDSUSERDllExport extern PRESTypePluginKeyKind
    HelloWorldPlugin_get_key_kind(void);
    NDDSUSERDllExport extern unsigned int
    HelloWorldPlugin_get_serialized_key_max_size(
        PRESTypePluginEndpointData endpoint_data,
        RTIBool include_encapsulation,
        RTIEncapsulationId encapsulation_id,
        unsigned int current_alignment);
    NDDSUSERDllExport extern unsigned int
    HelloWorldPlugin_get_serialized_key_max_size_for_keyhash(
        PRESTypePluginEndpointData endpoint_data,
        RTIEncapsulationId encapsulation_id,
        unsigned int current_alignment);
    NDDSUSERDllExport extern RTIBool
```

```
    HelloWorldPlugin_deserialize_key(
        PRESTypePluginEndpointData endpoint_data,
        HelloWorld ** sample,
        RTIBool * drop_sample,
        struct RTICdrStream *cdrStream,
        RTIBool deserialize_encapsulation,
        RTIBool deserialize_key,
        void *endpoint_plugin_qos);
    NDDSUSERDllExport extern
    struct RTIXCdrInterpreterPrograms * HelloWorldPlugin_get_programs(void);
    /* Plugin Functions */
    NDDSUSERDllExport extern struct PRESTypePlugin*
    HelloWorldPlugin_new(void);
    NDDSUSERDllExport extern void
    HelloWorldPlugin_delete(struct PRESTypePlugin *);
    #ifdef __cplusplus
}
#endif
#if defined(NDDS_USER_DLL_EXPORT) || defined(NDDS_USER_SYMBOL_EXPORT)
#undef NDDSUSERDllExport
#define NDDSUSERDllExport
#endif
#endif /* HelloWorldPlugin_1436885481_h */
```

[$(NDDSHOME)/example/CPP/helloWorldPersistence/HelloWorldPlugin.cxx]

```
/*
WARNING: THIS FILE IS AUTO-GENERATED. DO NOT MODIFY.

This file was generated from HelloWorld.idl
using RTI Code Generator (rtiddsgen) version 4.7.0.
The rtiddsgen tool is part of the RTI Connext DDS distribution.
For more information, type 'rtiddsgen -help' at a command shell
or consult the Code Generator User's Manual.
*/
#include <string.h>
#ifndef ndds_cpp_h
#include "ndds/ndds_cpp.h"
#endif
#ifndef cdr_type_h
#include "cdr/cdr_type.h"
#endif
#ifndef cdr_type_object_h
#include "cdr/cdr_typeObject.h"
#endif
#ifndef cdr_encapsulation_h
#include "cdr/cdr_encapsulation.h"
#endif
#ifndef cdr_stream_h
#include "cdr/cdr_stream.h"
#endif
#include "xcdr/xcdr_interpreter.h"
#include "xcdr/xcdr_stream.h"
#ifndef cdr_log_h
#include "cdr/cdr_log.h"
#endif
#ifndef pres_typePlugin_h
#include "pres/pres_typePlugin.h"
#endif
#include "dds_c/dds_c_typecode_impl.h"
#define RTI_CDR_CURRENT_SUBMODULE RTI_CDR_SUBMODULE_MASK_STREAM
#include <new>
#include "HelloWorldPlugin.h"
/* ------------------------------------------------------------------------
*  Type HelloWorld
* ------------------------------------------------------------------------ */
/* ----------------------------------------------------------------------------
Support functions:
* ------------------------------------------------------------------------ */
HelloWorld*
HelloWorldPluginSupport_create_data_w_params(
    const struct DDS_TypeAllocationParams_t * alloc_params)
{
    HelloWorld *sample = NULL;
    if (alloc_params == NULL) {
        return NULL;
    } else if(!alloc_params->allocate_memory) {
        RTICdrLog_exception(&RTI_CDR_LOG_TYPE_OBJECT_NOT_ASSIGNABLE_ss,
        "alloc_params->allocate_memory","false");
        return NULL;
    }
```

```
    sample = new (std::nothrow) HelloWorld();
    if (sample == NULL) {
        return NULL;
    }
    if (!HelloWorld_initialize_w_params(sample,alloc_params)) {
        struct DDS_TypeDeallocationParams_t deallocParams =
        DDS_TYPE_DEALLOCATION_PARAMS_DEFAULT;
        deallocParams.delete_pointers = alloc_params->allocate_pointers;
        deallocParams.delete_optional_members = alloc_params->allocate_pointers;
        /* Coverity reports a possible uninit_use_in_call that will happen if the
        allocation fails. But if the allocation fails then sample == null and
        the method will return before reach this point.*/
        /* Coverity reports a possible overwrite_var on the members of the sample.
        It is a false positive since all the pointers are freed before assigning
        null to them. */
        /* coverity[uninit_use_in_call : FALSE] */
        /* coverity[overwrite_var : FALSE] */
        HelloWorld_finalize_w_params(sample, &deallocParams);
        /* Coverity reports a possible leaked_storage on the sample members when
        freeing sample. It is a false positive since all the members' memory
        is freed in the call "HelloWorld_finalize_ex" */
        /* coverity[leaked_storage : FALSE] */
        delete  sample;
        sample=NULL;
    }
    return sample;
}
HelloWorld *
HelloWorldPluginSupport_create_data_ex(RTIBool allocate_pointers)
{
    HelloWorld *sample = NULL;
    sample = new (std::nothrow) HelloWorld();
    if(sample == NULL) {
        return NULL;
    }
    /* coverity[example_checked : FALSE] */
    if (!HelloWorld_initialize_ex(sample,allocate_pointers, RTI_TRUE)) {
        /* Coverity reports a possible uninit_use_in_call that will happen if the
        new fails. But if new fails then sample == null and the method will
        return before reach this point. */
        /* Coverity reports a possible overwrite_var on the members of the sample.
        It is a false positive since all the pointers are freed before assigning
        null to them. */
        /* coverity[uninit_use_in_call : FALSE] */
        /* coverity[overwrite_var : FALSE] */
        HelloWorld_finalize_ex(sample, RTI_TRUE);
        /* Coverity reports a possible leaked_storage on the sample members when
        freeing sample. It is a false positive since all the members' memory
        is freed in the call "HelloWorld_finalize_ex" */
        /* coverity[leaked_storage : FALSE] */
        delete  sample;
        sample=NULL;
    }
    return sample;
}
void *
HelloWorldPluginSupport_create_dataI(void)
{
    return HelloWorldPluginSupport_create_data_ex(RTI_TRUE);
}
HelloWorld *
HelloWorldPluginSupport_create_data(void)
{
    return (HelloWorld *) HelloWorldPluginSupport_create_dataI();
}
void
HelloWorldPluginSupport_destroy_data_w_params(
    HelloWorld *sample,
    const struct DDS_TypeDeallocationParams_t * dealloc_params) {
    HelloWorld_finalize_w_params(sample,dealloc_params);
    delete  sample;
}
void
HelloWorldPluginSupport_destroy_data_ex(
    HelloWorld *sample,RTIBool deallocate_pointers) {
    HelloWorld_finalize_ex(sample,deallocate_pointers);
    delete  sample;
}
void
HelloWorldPluginSupport_destroy_data(
    HelloWorld *sample) {
```

```
    HelloWorldPluginSupport_destroy_data_ex(sample,RTI_TRUE);
}
void
HelloWorldPluginSupport_destroy_dataI(
    void *sample)
{
    HelloWorldPluginSupport_destroy_data((HelloWorld *) sample);
}
RTIBool
HelloWorldPluginSupport_copy_data(
    HelloWorld *dst,
    const HelloWorld *src)
{
    return HelloWorld_copy(dst,(const HelloWorld*) src);
}
void
HelloWorldPluginSupport_print_data(
    const HelloWorld *sample,
    const char *desc,
    unsigned int indent_level)
{
    RTICdrType_printIndent(indent_level);
    if (desc != NULL) {
        RTILogParamString_printPlain("%s:\n", desc);
    } else {
        RTILogParamString_printPlain("\n");
    }
    if (sample == NULL) {
        RTILogParamString_printPlain("NULL\n");
        return;
    }
    RTICdrType_printLong(
        &sample->data,
        "data",
        RTIOsapiUtility_uInt32Plus1(indent_level));
}
/* ------------------------------------------------------------------------
Callback functions:
* ------------------------------------------------------------------------ */
PRESTypePluginParticipantData
HelloWorldPlugin_on_participant_attached(
    void *registration_data,
    const struct PRESTypePluginParticipantInfo *participant_info,
    RTIBool top_level_registration,
    void *container_plugin_context,
    RTICdrTypeCode *type_code)
{
    struct RTIXCdrInterpreterPrograms *programs = NULL;
    struct RTIXCdrInterpreterProgramsGenProperty programProperty =
    RTIXCdrInterpreterProgramsGenProperty_INITIALIZER;
    struct PRESTypePluginDefaultParticipantData *pd = NULL;
    RTIOsapiUtility_unusedParameter(registration_data);
    RTIOsapiUtility_unusedParameter(participant_info);
    RTIOsapiUtility_unusedParameter(top_level_registration);
    RTIOsapiUtility_unusedParameter(container_plugin_context);
    RTIOsapiUtility_unusedParameter(type_code);
    pd = (struct PRESTypePluginDefaultParticipantData *)
    PRESTypePluginDefaultParticipantData_new(participant_info);
    programProperty.generateV1Encapsulation = RTI_XCDR_TRUE;
    programProperty.generateV2Encapsulation = RTI_XCDR_TRUE;
    programProperty.resolveAlias = RTI_XCDR_TRUE;
    programProperty.inlineStruct = RTI_XCDR_TRUE;
    programProperty.optimizeEnum = RTI_XCDR_TRUE;
    programProperty.unboundedSize = RTIXCdrLong_MAX;
    programs = DDS_TypeCodeFactory_assert_programs_in_global_list(
        DDS_TypeCodeFactory_get_instance(),
        HelloWorld_get_typecode(),
        &programProperty,
        RTI_XCDR_PROGRAM_MASK_TYPEPLUGIN);
    if (programs == NULL) {
        PRESTypePluginDefaultParticipantData_delete(
            (PRESTypePluginParticipantData) pd);
        return NULL;
    }
    pd->programs = programs;
    return (PRESTypePluginParticipantData)pd;
}
void
HelloWorldPlugin_on_participant_detached(
    PRESTypePluginParticipantData participant_data)
{
```

```
    if (participant_data != NULL) {
        struct PRESTypePluginDefaultParticipantData *pd =
        (struct PRESTypePluginDefaultParticipantData *)participant_data;
        if (pd->programs != NULL) {
            DDS_TypeCodeFactory_remove_programs_from_global_list(
                DDS_TypeCodeFactory_get_instance(),
                pd->programs);
            pd->programs = NULL;
        }
        PRESTypePluginDefaultParticipantData_delete(participant_data);
    }
}
PRESTypePluginEndpointData
HelloWorldPlugin_on_endpoint_attached(
    PRESTypePluginParticipantData participant_data,
    const struct PRESTypePluginEndpointInfo *endpoint_info,
    RTIBool top_level_registration,
    void *containerPluginContext)
{
    PRESTypePluginEndpointData epd = NULL;
    unsigned int serializedSampleMaxSize = 0;
    RTIOsapiUtility_unusedParameter(top_level_registration);
    RTIOsapiUtility_unusedParameter(containerPluginContext);
    if (participant_data == NULL) {
        return NULL;
    }
    epd = PRESTypePluginDefaultEndpointData_new(
        participant_data,
        endpoint_info,
        HelloWorldPluginSupport_create_dataI,
        HelloWorldPluginSupport_destroy_dataI,
        NULL , NULL );
    if (epd == NULL) {
        return NULL;
    }
    if (endpoint_info->endpointKind == PRES_TYPEPLUGIN_ENDPOINT_WRITER) {
        serializedSampleMaxSize = HelloWorldPlugin_get_serialized_sample_max_size(
            epd,RTI_FALSE,RTI_CDR_ENCAPSULATION_ID_CDR_BE,0);
        PRESTypePluginDefaultEndpointData_setMaxSizeSerializedSample(epd, serializedSampleMaxSize);
        if (PRESTypePluginDefaultEndpointData_createWriterPool(
            epd,
            endpoint_info,
            (PRESTypePluginGetSerializedSampleMaxSizeFunction)
            HelloWorldPlugin_get_serialized_sample_max_size, epd,
            (PRESTypePluginGetSerializedSampleSizeFunction)
            PRESTypePlugin_interpretedGetSerializedSampleSize,
            epd) == RTI_FALSE) {
            PRESTypePluginDefaultEndpointData_delete(epd);
            return NULL;
        }
    }
    return epd;
}
void
HelloWorldPlugin_on_endpoint_detached(
    PRESTypePluginEndpointData endpoint_data)
{
    PRESTypePluginDefaultEndpointData_delete(endpoint_data);
}
void
HelloWorldPlugin_return_sample(
    PRESTypePluginEndpointData endpoint_data,
    HelloWorld *sample,
    void *handle)
{
    HelloWorld_finalize_optional_members(sample, RTI_TRUE);
    PRESTypePluginDefaultEndpointData_returnSample(
        endpoint_data, sample, handle);
}
void HelloWorldPlugin_finalize_optional_members(
    PRESTypePluginEndpointData endpoint_data,
    HelloWorld* sample,
    RTIBool deletePointers)
{
    RTIOsapiUtility_unusedParameter(endpoint_data);
    HelloWorld_finalize_optional_members(
        sample, deletePointers);
}
RTIBool
HelloWorldPlugin_copy_sample(
    PRESTypePluginEndpointData endpoint_data,
```

```
    HelloWorld *dst,
    const HelloWorld *src)
{
    RTIOsapiUtility_unusedParameter(endpoint_data);
    return HelloWorldPluginSupport_copy_data(dst,src);
}
/* ------------------------------------------------------------------------
(De)Serialize functions:
* ------------------------------------------------------------------------ */
unsigned int
HelloWorldPlugin_get_serialized_sample_max_size(
    PRESTypePluginEndpointData endpoint_data,
    RTIBool include_encapsulation,
    RTIEncapsulationId encapsulation_id,
    unsigned int current_alignment);
RTIBool
HelloWorldPlugin_serialize_to_cdr_buffer_ex(
    char *buffer,
    unsigned int *length,
    const HelloWorld *sample,
    DDS_DataRepresentationId_t representation)
{
    RTIEncapsulationId encapsulationId = RTI_CDR_ENCAPSULATION_ID_INVALID;
    struct RTICdrStream cdrStream;
    struct PRESTypePluginDefaultEndpointData epd;
    RTIBool result;
    struct PRESTypePluginDefaultParticipantData pd;
    struct RTIXCdrTypePluginProgramContext defaultProgramContext =
    RTIXCdrTypePluginProgramContext_INTIALIZER;
    struct PRESTypePlugin plugin;
    if (length == NULL) {
        return RTI_FALSE;
    }
    RTIOsapiMemory_zero(&epd, sizeof(struct PRESTypePluginDefaultEndpointData));
    epd.programContext = defaultProgramContext;
    epd._participantData = &pd;
    epd.typePlugin = &plugin;
    epd.programContext.endpointPluginData = &epd;
    plugin.typeCode = (struct RTICdrTypeCode *)
    HelloWorld_get_typecode();
    pd.programs = HelloWorldPlugin_get_programs();
    if (pd.programs == NULL) {
        return RTI_FALSE;
    }
    encapsulationId = DDS_TypeCode_get_native_encapsulation(
        (DDS_TypeCode *) plugin.typeCode,
        representation);
    if (encapsulationId == RTI_CDR_ENCAPSULATION_ID_INVALID) {
        return RTI_FALSE;
    }
    epd._maxSizeSerializedSample =
    HelloWorldPlugin_get_serialized_sample_max_size(
        (PRESTypePluginEndpointData)&epd,
        RTI_TRUE,
        encapsulationId,
        0);
    if (buffer == NULL) {
        *length =
        PRESTypePlugin_interpretedGetSerializedSampleSize(
            (PRESTypePluginEndpointData)&epd,
            RTI_TRUE,
            encapsulationId,
            0,
            sample);
        if (*length == 0) {
            return RTI_FALSE;
        }
        return RTI_TRUE;
    }
    RTICdrStream_init(&cdrStream);
    RTICdrStream_set(&cdrStream, (char *)buffer, *length);
    result = PRESTypePlugin_interpretedSerialize(
        (PRESTypePluginEndpointData)&epd,
        sample,
        &cdrStream,
        RTI_TRUE,
        encapsulationId,
        RTI_TRUE,
        NULL);
    *length = (unsigned int) RTICdrStream_getCurrentPositionOffset(&cdrStream);
    return result;
```

```
}
RTIBool
HelloWorldPlugin_serialize_to_cdr_buffer(
    char *buffer,
    unsigned int *length,
    const HelloWorld *sample)
{
    return HelloWorldPlugin_serialize_to_cdr_buffer_ex(
        buffer,
        length,
        sample,
        DDS_AUTO_DATA_REPRESENTATION);
}
RTIBool
HelloWorldPlugin_deserialize_from_cdr_buffer(
    HelloWorld *sample,
    const char * buffer,
    unsigned int length)
{
    struct RTICdrStream cdrStream;
    struct PRESTypePluginDefaultEndpointData epd;
    struct RTIXCdrTypePluginProgramContext defaultProgramContext =
    RTIXCdrTypePluginProgramContext_INTIALIZER;
    struct PRESTypePluginDefaultParticipantData pd;
    struct PRESTypePlugin plugin;
    epd.programContext = defaultProgramContext;
    epd._participantData = &pd;
    epd.typePlugin = &plugin;
    epd.programContext.endpointPluginData = &epd;
    plugin.typeCode = (struct RTICdrTypeCode *)
    HelloWorld_get_typecode();
    pd.programs = HelloWorldPlugin_get_programs();
    if (pd.programs == NULL) {
        return RTI_FALSE;
    }
    RTIXCdrSampleAssignabilityProperty_setFromGlobalComplianceMask(
        &epd._assignabilityProperty);
    RTICdrStream_init(&cdrStream);
    RTICdrStream_set(&cdrStream, (char *)buffer, length);
    HelloWorld_finalize_optional_members(sample, RTI_TRUE);
    return PRESTypePlugin_interpretedDeserialize(
        (PRESTypePluginEndpointData)&epd, sample,
        &cdrStream, RTI_TRUE, RTI_TRUE,
        NULL);
}
#if !defined(NDDS_STANDALONE_TYPE)
DDS_ReturnCode_t
HelloWorldPlugin_data_to_string(
    const HelloWorld *sample,
    char *_str,
    DDS_UnsignedLong *str_size,
    const struct DDS_PrintFormatProperty *property)
{
    DDS_DynamicData *data = NULL;
    char *buffer = NULL;
    unsigned int length = 0;
    struct DDS_PrintFormat printFormat;
    DDS_ReturnCode_t retCode = DDS_RETCODE_ERROR;
    if (sample == NULL) {
        return DDS_RETCODE_BAD_PARAMETER;
    }
    if (str_size == NULL) {
        return DDS_RETCODE_BAD_PARAMETER;
    }
    if (property == NULL) {
        return DDS_RETCODE_BAD_PARAMETER;
    }
    if (!HelloWorldPlugin_serialize_to_cdr_buffer(
        NULL,
        &length,
        sample)) {
        return DDS_RETCODE_ERROR;
    }
    RTIOsapiHeap_allocateBuffer(&buffer, length, RTI_OSAPI_ALIGNMENT_DEFAULT);
    if (buffer == NULL) {
        return DDS_RETCODE_ERROR;
    }
    if (!HelloWorldPlugin_serialize_to_cdr_buffer(
        buffer,
        &length,
        sample)) {
```

```
        RTIOsapiHeap_freeBuffer(buffer);
        return DDS_RETCODE_ERROR;
    }
    data = DDS_DynamicData_new(
        HelloWorld_get_typecode(),
        &DDS_DYNAMIC_DATA_PROPERTY_DEFAULT);
    if (data == NULL) {
        RTIOsapiHeap_freeBuffer(buffer);
        return DDS_RETCODE_ERROR;
    }
    retCode = DDS_DynamicData_from_cdr_buffer(data, buffer, length);
    if (retCode != DDS_RETCODE_OK) {
        RTIOsapiHeap_freeBuffer(buffer);
        DDS_DynamicData_delete(data);
        return retCode;
    }
    retCode = DDS_PrintFormatProperty_to_print_format(
        property,
        &printFormat);
    if (retCode != DDS_RETCODE_OK) {
        RTIOsapiHeap_freeBuffer(buffer);
        DDS_DynamicData_delete(data);
        return retCode;
    }
    retCode = DDS_DynamicDataFormatter_to_string_w_format(
        data,
        _str,
        str_size,
        &printFormat);
    if (retCode != DDS_RETCODE_OK) {
        RTIOsapiHeap_freeBuffer(buffer);
        DDS_DynamicData_delete(data);
        return retCode;
    }
    RTIOsapiHeap_freeBuffer(buffer);
    DDS_DynamicData_delete(data);
    return DDS_RETCODE_OK;
}
#endif
unsigned int
HelloWorldPlugin_get_serialized_sample_max_size(
    PRESTypePluginEndpointData endpoint_data,
    RTIBool include_encapsulation,
    RTIEncapsulationId encapsulation_id,
    unsigned int current_alignment)
{
    unsigned int size;
    RTIBool overflow = RTI_FALSE;
    size = PRESTypePlugin_interpretedGetSerializedSampleMaxSize(
        endpoint_data,&overflow,include_encapsulation,encapsulation_id,current_alignment);
    if (overflow) {
        size = RTI_CDR_MAX_SERIALIZED_SIZE;
    }
    return size;
}
/* ----------------------------------------------------------------------------------------
Key Management functions:
* ---------------------------------------------------------------------------------------- */
PRESTypePluginKeyKind
HelloWorldPlugin_get_key_kind(void)
{
    return PRES_TYPEPLUGIN_NO_KEY;
}
RTIBool HelloWorldPlugin_deserialize_key(
    PRESTypePluginEndpointData endpoint_data,
    HelloWorld **sample,
    RTIBool * drop_sample,
    struct RTICdrStream *cdrStream,
    RTIBool deserialize_encapsulation,
    RTIBool deserialize_key,
    void *endpoint_plugin_qos)
{
    RTIBool result;
    RTIOsapiUtility_unusedParameter(drop_sample);
    /*  Depending on the type and the flags used in rtiddsgen, coverity may detect
    that sample is always null. Since the case is very dependant on
    the IDL/XML and the configuration we keep the check for safety.
    */
    result= PRESTypePlugin_interpretedDeserializeKey(
        endpoint_data,
        /* coverity[check_after_deref] */
```

```
        (sample != NULL) ? *sample : NULL,
        cdrStream,
        deserialize_encapsulation,
        deserialize_key,
        endpoint_plugin_qos);
    return result;
}
unsigned int
HelloWorldPlugin_get_serialized_key_max_size(
    PRESTypePluginEndpointData endpoint_data,
    RTIBool include_encapsulation,
    RTIEncapsulationId encapsulation_id,
    unsigned int current_alignment)
{
    unsigned int size;
    RTIBool overflow = RTI_FALSE;
    size = PRESTypePlugin_interpretedGetSerializedKeyMaxSize(
        endpoint_data,&overflow,include_encapsulation,encapsulation_id,current_alignment);
    if (overflow) {
        size = RTI_CDR_MAX_SERIALIZED_SIZE;
    }
    return size;
}
unsigned int
HelloWorldPlugin_get_serialized_key_max_size_for_keyhash(
    PRESTypePluginEndpointData endpoint_data,
    RTIEncapsulationId encapsulation_id,
    unsigned int current_alignment)
{
    unsigned int size;
    RTIBool overflow = RTI_FALSE;
    size = PRESTypePlugin_interpretedGetSerializedKeyMaxSizeForKeyhash(
        endpoint_data,
        &overflow,
        encapsulation_id,
        current_alignment);
    if (overflow) {
        size = RTI_CDR_MAX_SERIALIZED_SIZE;
    }
    return size;
}
struct RTIXCdrInterpreterPrograms * HelloWorldPlugin_get_programs(void)
{
    return ::rti::xcdr::get_cdr_serialization_programs<
    HelloWorld,
    true, true, true>();
}
/* ------------------------------------------------------------------------
* Plug-in Installation Methods
* ------------------------------------------------------------------------ */
struct PRESTypePlugin *HelloWorldPlugin_new(void)
{
    struct PRESTypePlugin *plugin = NULL;
    const struct PRESTypePluginVersion PLUGIN_VERSION =
    PRES_TYPE_PLUGIN_VERSION_2_0;
    RTIOsapiHeap_allocateStructure(
        &plugin, struct PRESTypePlugin);
    if (plugin == NULL) {
        return NULL;
    }
    plugin->version = PLUGIN_VERSION;
    /* set up parent's function pointers */
    plugin->onParticipantAttached =
    (PRESTypePluginOnParticipantAttachedCallback)
    HelloWorldPlugin_on_participant_attached;
    plugin->onParticipantDetached =
    (PRESTypePluginOnParticipantDetachedCallback)
    HelloWorldPlugin_on_participant_detached;
    plugin->onEndpointAttached =
    (PRESTypePluginOnEndpointAttachedCallback)
    HelloWorldPlugin_on_endpoint_attached;
    plugin->onEndpointDetached =
    (PRESTypePluginOnEndpointDetachedCallback)
    HelloWorldPlugin_on_endpoint_detached;
    plugin->copySampleFnc =
    (PRESTypePluginCopySampleFunction)
    HelloWorldPlugin_copy_sample;
    plugin->createSampleFnc =
    (PRESTypePluginCreateSampleFunction)
    HelloWorldPlugin_create_sample;
    plugin->destroySampleFnc =
```

```
    (PRESTypePluginDestroySampleFunction)
    HelloWorldPlugin_destroy_sample;
    plugin->finalizeOptionalMembersFnc =
    (PRESTypePluginFinalizeOptionalMembersFunction)
    HelloWorldPlugin_finalize_optional_members;
    plugin->serializeFnc =
    (PRESTypePluginSerializeFunction) PRESTypePlugin_interpretedSerialize;
    plugin->deserializeFnc =
    (PRESTypePluginDeserializeFunction) PRESTypePlugin_interpretedDeserializeWithAlloc;
    plugin->getSerializedSampleMaxSizeFnc =
    (PRESTypePluginGetSerializedSampleMaxSizeFunction)
    HelloWorldPlugin_get_serialized_sample_max_size;
    plugin->getSerializedSampleMinSizeFnc =
    (PRESTypePluginGetSerializedSampleMinSizeFunction)
    PRESTypePlugin_interpretedGetSerializedSampleMinSize;
    plugin->getDeserializedSampleMaxSizeFnc = NULL;
    plugin->getSampleFnc =
    (PRESTypePluginGetSampleFunction)
    HelloWorldPlugin_get_sample;
    plugin->returnSampleFnc =
    (PRESTypePluginReturnSampleFunction)
    HelloWorldPlugin_return_sample;
    plugin->getKeyKindFnc =
    (PRESTypePluginGetKeyKindFunction)
    HelloWorldPlugin_get_key_kind;
    /* These functions are only used for keyed types. As this is not a keyed
    type they are all set to NULL
    */
    plugin->serializeKeyFnc = NULL ;
    plugin->deserializeKeyFnc = NULL;
    plugin->getKeyFnc = NULL;
    plugin->returnKeyFnc = NULL;
    plugin->instanceToKeyFnc = NULL;
    plugin->keyToInstanceFnc = NULL;
    plugin->getSerializedKeyMaxSizeFnc = NULL;
    plugin->instanceToKeyHashFnc = NULL;
    plugin->serializedSampleToKeyHashFnc = NULL;
    plugin->serializedKeyToKeyHashFnc = NULL;
    #ifdef NDDS_STANDALONE_TYPE
    plugin->typeCode = NULL;
    #else
    plugin->typeCode =  (struct RTICdrTypeCode *)HelloWorld_get_typecode();
    #endif
    plugin->languageKind = PRES_TYPEPLUGIN_CPP_LANG;
    /* Serialized buffer */
    plugin->getBuffer =
    (PRESTypePluginGetBufferFunction)
    HelloWorldPlugin_get_buffer;
    plugin->returnBuffer =
    (PRESTypePluginReturnBufferFunction)
    HelloWorldPlugin_return_buffer;
    plugin->getBufferWithParams = NULL;
    plugin->returnBufferWithParams = NULL;
    plugin->getSerializedSampleSizeFnc =
    (PRESTypePluginGetSerializedSampleSizeFunction)
    PRESTypePlugin_interpretedGetSerializedSampleSize;
    plugin->getWriterLoanedSampleFnc = NULL;
    plugin->returnWriterLoanedSampleFnc = NULL;
    plugin->returnWriterLoanedSampleFromCookieFnc = NULL;
    plugin->validateWriterLoanedSampleFnc = NULL;
    plugin->setWriterLoanedSampleSerializedStateFnc = NULL;
    plugin->endpointTypeName = HelloWorldTYPENAME;
    plugin->isMetpType = RTI_FALSE;
    plugin->isRecursiveType = RTI_FALSE;
    return plugin;
}
void
HelloWorldPlugin_delete(struct PRESTypePlugin *plugin)
{
    RTIOsapiHeap_freeStructure(plugin);
}
#undef RTI_CDR_CURRENT_SUBMODULE
```

5.17 HelloWorldSupport.cxx

[$(NDDSHOME)/example/CPP/helloWorldPersistence/HelloWorldSupport.h]

```
/*
WARNING: THIS FILE IS AUTO-GENERATED. DO NOT MODIFY.

This file was generated from HelloWorld.idl
using RTI Code Generator (rtiddsgen) version 4.7.0.
The rtiddsgen tool is part of the RTI Connext DDS distribution.
For more information, type 'rtiddsgen -help' at a command shell
or consult the Code Generator User's Manual.
*/
#ifndef HelloWorldSupport_1436885481_h
#define HelloWorldSupport_1436885481_h
/* Uses */
#include "HelloWorld.h"
#ifndef ndds_c_h
#include "ndds/ndds_c.h"
#endif
#ifdef __cplusplus
extern "C" {
    #endif
    #if (defined(RTI_WIN32) || defined (RTI_WINCE) || defined(RTI_INTIME)) && defined(NDDS_USER_DLL_EXPORT)
    #endif
    /* ========================================================================= */
    #if defined(NDDS_USER_DLL_EXPORT) && defined(RTI_WIN32)
    #undef NDDSUSERDllExport
    #define NDDSUSERDllExport __declspec(dllexport)
    #endif
    #if !defined(RTI_WIN32) && defined(NDDS_USER_SYMBOL_EXPORT)
    #undef NDDSUSERDllExport
    #define NDDSUSERDllExport __attribute__((visibility("default")))
    #endif
    DDS_TYPESUPPORT_C(HelloWorldTypeSupport, HelloWorld);
    DDS_DATAWRITER_WITH_DATA_CONSTRUCTOR_METHODS_C(HelloWorldDataWriter, HelloWorld);
    DDS_DATAREADER_C(HelloWorldDataReader, HelloWorldSeq, HelloWorld);
    #if defined(NDDS_USER_DLL_EXPORT) || defined(NDDS_USER_SYMBOL_EXPORT)
    #undef NDDSUSERDllExport
    #define NDDSUSERDllExport
    #endif
    #ifdef __cplusplus
}
#endif
#endif  /* HelloWorldSupport_1436885481_h */
```

[$(NDDSHOME)/example/CPP/helloWorldPersistence/HelloWorldSupport.cxx]

```
/*
WARNING: THIS FILE IS AUTO-GENERATED. DO NOT MODIFY.

This file was generated from HelloWorld.idl
using RTI Code Generator (rtiddsgen) version 4.7.0.
The rtiddsgen tool is part of the RTI Connext DDS distribution.
For more information, type 'rtiddsgen -help' at a command shell
or consult the Code Generator User's Manual.
*/
#include "HelloWorldSupport.h"
#include "HelloWorldPlugin.h"
#ifndef dds_c_log_impl_h
#include "dds_c/dds_c_log_impl.h"
#endif
/* ========================================================================= */
/* ------------------------------------------------------------------ */
/* DDSDataWriter
*/
/* Requires */
#define TTYPENAME   HelloWorldTYPENAME
/* Defines */
#define TDataWriter HelloWorldDataWriter
#define TData       HelloWorld
#define RTI_ENABLE_TDATAWRITER_DATA_CONSTRUCTOR_METHODS
#include "dds_cpp/generic/dds_cpp_data_TDataWriter.gen"
#undef RTI_ENABLE_TDATAWRITER_DATA_CONSTRUCTOR_METHODS
#undef TDataWriter
#undef TData
#undef TTYPENAME
/* ------------------------------------------------------------------ */
/* DDSDataReader
*/
/* Requires */
#define TTYPENAME   HelloWorldTYPENAME
/* Defines */
#define TDataReader HelloWorldDataReader
#define TDataSeq    HelloWorldSeq
```

```
#define TData       HelloWorld
#define RTI_ENABLE_TDATAREADER_DATA_CONSISTENCY_CHECK_METHOD
#include "dds_cpp/generic/dds_cpp_data_TDataReader.gen"
#undef RTI_ENABLE_TDATAREADER_DATA_CONSISTENCY_CHECK_METHOD
#undef TDataReader
#undef TDataSeq
#undef TData
#undef TTYPENAME
/* ------------------------------------------------------------------ */
/* TypeSupport

«IMPLEMENTATION »

Requires:  TTYPENAME,
TPlugin_new
TPlugin_delete
Defines:   TTypeSupport, TData, TDataReader, TDataWriter
*/
/* Requires */
#define TTYPENAME    HelloWorldTYPENAME
#define TPlugin_new  HelloWorldPlugin_new
#define TPlugin_delete  HelloWorldPlugin_delete
/* Defines */
#define TTypeSupport HelloWorldTypeSupport
#define TData        HelloWorld
#define TDataReader  HelloWorldDataReader
#define TDataWriter  HelloWorldDataWriter
#define TGENERATE_SER_CODE
#ifndef NDDS_STANDALONE_TYPE
#define TGENERATE_TYPECODE
#endif
#include "dds_cpp/generic/dds_cpp_data_TTypeSupport.gen"
#undef TTypeSupport
#undef TData
#undef TDataReader
#undef TDataWriter
#ifndef NDDS_STANDALONE_TYPE
#undef TGENERATE_TYPECODE
#endif
#undef TGENERATE_SER_CODE
#undef TTYPENAME
#undef TPlugin_new
#undef TPlugin_delete
```

5.18 Java Example

Persistence Example Using Java.

Modules

- **HelloWorld.idl**
- **HelloWorld.java**
- **HelloWorldDataReader.java**
- **HelloWorldDataWriter.java**
- **HelloWorldPublisher.java**
- **HelloWorldSubscriber.java**
- **HelloWorldSeq.java**
- **HelloWorldTypeCode.java**
- **HelloWorldTypeSupport.java**

5.18.1 Detailed Description

Persistence Example Using Java.

See README.txt for details.

[$(NDDSHOME)/example/JAVA/helloWorldPersistence/README.txt]

5.19 HelloWorld.idl

[$(NDDSHOME)/example/JAVA/helloWorldPersistence/HelloWorld.idl]

```
struct HelloWorld {
    long data;
};
```

5.20 HelloWorld.java

[$(NDDSHOME)/example/JAVA/helloWorldPersistence/HelloWorld.java]

```
/*
WARNING: THIS FILE IS AUTO-GENERATED. DO NOT MODIFY.

This file was generated from .idl
using RTI Code Generator (rtiddsgen) version 4.7.0.
The rtiddsgen tool is part of the RTI Connext DDS distribution.
For more information, type 'rtiddsgen -help' at a command shell
or consult the Code Generator User's Manual.
*/
import com.rti.dds.infrastructure.*;
import com.rti.dds.infrastructure.Copyable;
import java.io.Serializable;
import com.rti.dds.cdr.CdrHelper;
// Depending on the type represented in the IDL, we may perform some redundant
// casts, we are suppressing that warning
@SuppressWarnings("cast")
public class HelloWorld   implements Copyable, Serializable{
    private static final long serialVersionUID = -1545509471L;
    public int data = (int)0;
    public HelloWorld() {
    }
    public HelloWorld (HelloWorld other) {
        this();
        copy_from(other);
    }
    public static java.lang.Object create() {
        HelloWorld self;
        self = new  HelloWorld();
        self.clear();
        return self;
    }
    public void clear() {
        data = (int)0;
    }
    @Override
    public boolean equals(java.lang.Object o) {
        if (o == null) {
            return false;
        }
        if(getClass() != o.getClass()) {
            return false;
        }
        HelloWorld otherObj = (HelloWorld)o;
        if(this.data != otherObj.data) {
            return false;
        }
        return true;
```

```
    }
    @Override
    public int hashCode() {
        final int __prime = 31;
        int __result = 1;
        __result = __prime * __result + (int)data;
        return __result;
    }
    public java.lang.Object copy_from(java.lang.Object src) {
        HelloWorld typedSrc = (HelloWorld) src;
        HelloWorld typedDst = this;
        typedDst.data = typedSrc.data;
        return this;
    }
    @Override
    public java.lang.String toString(){
        return toString("", 0);
    }
    public java.lang.String toString(java.lang.String desc, int indent) {
        java.lang.StringBuffer strBuffer = new java.lang.StringBuffer();
        if (desc != null) {
            CdrHelper.printIndent(strBuffer, indent);
            strBuffer.append(desc).append(":\n");
        }
        CdrHelper.printIndent(strBuffer, indent+1);
        strBuffer.append("data: ").append(this.data).append("\n");
        return strBuffer.toString();
    }
}
```

5.21 HelloWorldDataReader.java

[$(NDDSHOME)/example/JAVA/helloWorldPersistence/HelloWorldDataReader.java]

```
/*
WARNING: THIS FILE IS AUTO-GENERATED. DO NOT MODIFY.

This file was generated from .idl
using RTI Code Generator (rtiddsgen) version 4.7.0.
The rtiddsgen tool is part of the RTI Connext DDS distribution.
For more information, type 'rtiddsgen -help' at a command shell
or consult the Code Generator User's Manual.
*/
import com.rti.dds.infrastructure.InstanceHandle_t;
import com.rti.dds.subscription.DataReaderImpl;
import com.rti.dds.subscription.DataReaderListener;
import com.rti.dds.subscription.ReadCondition;
import com.rti.dds.subscription.SampleInfo;
import com.rti.dds.subscription.SampleInfoSeq;
import com.rti.dds.topic.TypeSupportImpl;
// ===========================================================================
public class HelloWorldDataReader extends DataReaderImpl {
    // -----------------------------------------------------------------------
    // Public Methods
    // -----------------------------------------------------------------------
    public void read(HelloWorldSeq received_data, SampleInfoSeq info_seq,
    int max_samples,
    int sample_states, int view_states, int instance_states) {
        read_untyped(received_data, info_seq, max_samples, sample_states,
        view_states, instance_states);
    }
    public void take(HelloWorldSeq received_data, SampleInfoSeq info_seq,
    int max_samples,
    int sample_states, int view_states, int instance_states) {
        take_untyped(received_data, info_seq, max_samples, sample_states,
        view_states, instance_states);
    }
    public void read_w_condition(HelloWorldSeq received_data,
    SampleInfoSeq info_seq,
    int max_samples,
    ReadCondition condition) {
        read_w_condition_untyped(received_data, info_seq, max_samples,
        condition);
    }
    public void take_w_condition(HelloWorldSeq received_data,
    SampleInfoSeq info_seq,
```

```
    int max_samples,
    ReadCondition condition) {
        take_w_condition_untyped(received_data, info_seq, max_samples,
        condition);
    }
    public void read_next_sample(HelloWorld received_data, SampleInfo sample_info) {
        read_next_sample_untyped(received_data, sample_info);
    }
    public void take_next_sample(HelloWorld received_data, SampleInfo sample_info) {
        take_next_sample_untyped(received_data, sample_info);
    }
    public void read_instance(HelloWorldSeq received_data, SampleInfoSeq info_seq,
    int max_samples, InstanceHandle_t a_handle, int sample_states,
    int view_states, int instance_states) {
        read_instance_untyped(received_data, info_seq, max_samples, a_handle,
        sample_states, view_states, instance_states);
    }
    public void take_instance(HelloWorldSeq received_data, SampleInfoSeq info_seq,
    int max_samples, InstanceHandle_t a_handle, int sample_states,
    int view_states, int instance_states) {
        take_instance_untyped(received_data, info_seq, max_samples, a_handle,
        sample_states, view_states, instance_states);
    }
    public void read_instance_w_condition(HelloWorldSeq received_data,
    SampleInfoSeq info_seq, int max_samples,
    InstanceHandle_t a_handle, ReadCondition condition) {
        read_instance_w_condition_untyped(received_data, info_seq,
        max_samples, a_handle, condition);
    }
    public void take_instance_w_condition(HelloWorldSeq received_data,
    SampleInfoSeq info_seq, int max_samples,
    InstanceHandle_t a_handle, ReadCondition condition) {
        take_instance_w_condition_untyped(received_data, info_seq,
        max_samples, a_handle, condition);
    }
    public void read_next_instance(HelloWorldSeq received_data,
    SampleInfoSeq info_seq, int max_samples,
    InstanceHandle_t a_handle, int sample_states, int view_states,
    int instance_states) {
        read_next_instance_untyped(received_data, info_seq, max_samples,
        a_handle, sample_states, view_states, instance_states);
    }
    public void take_next_instance(HelloWorldSeq received_data,
    SampleInfoSeq info_seq, int max_samples,
    InstanceHandle_t a_handle, int sample_states, int view_states,
    int instance_states) {
        take_next_instance_untyped(received_data, info_seq, max_samples,
        a_handle, sample_states, view_states, instance_states);
    }
    public void read_next_instance_w_condition(HelloWorldSeq received_data,
    SampleInfoSeq info_seq, int max_samples,
    InstanceHandle_t a_handle, ReadCondition condition) {
        read_next_instance_w_condition_untyped(received_data, info_seq,
        max_samples, a_handle, condition);
    }
    public void take_next_instance_w_condition(HelloWorldSeq received_data,
    SampleInfoSeq info_seq, int max_samples,
    InstanceHandle_t a_handle, ReadCondition condition) {
        take_next_instance_w_condition_untyped(received_data, info_seq,
        max_samples, a_handle, condition);
    }
    public void return_loan(HelloWorldSeq received_data, SampleInfoSeq info_seq) {
        return_loan_untyped(received_data, info_seq);
    }
    public void get_key_value(HelloWorld key_holder, InstanceHandle_t handle){
        get_key_value_untyped(key_holder, handle);
    }
    public InstanceHandle_t lookup_instance(HelloWorld key_holder) {
        return lookup_instance_untyped(key_holder);
    }
    // -----------------------------------------------------------------------
    // Package Methods
    // -----------------------------------------------------------------------
    // --- Constructors: -----------------------------------------------------
    /*package*/  HelloWorldDataReader (long native_reader, DataReaderListener listener,
    int mask, TypeSupportImpl data_type) {
        super(native_reader, listener, mask, data_type);
    }
}
```

5.22 HelloWorldDataWriter.java

[$(NDDSHOME)/example/JAVA/helloWorldPersistence/HelloWorldDataWriter.java]1

```
/*
WARNING: THIS FILE IS AUTO-GENERATED. DO NOT MODIFY.

This file was generated from .idl
using RTI Code Generator (rtiddsgen) version 4.7.0.
The rtiddsgen tool is part of the RTI Connext DDS distribution.
For more information, type 'rtiddsgen -help' at a command shell
or consult the Code Generator User's Manual.
*/
import com.rti.dds.infrastructure.Time_t;
import com.rti.dds.infrastructure.WriteParams_t;
import com.rti.dds.infrastructure.InstanceHandle_t;
import com.rti.dds.publication.DataWriterImpl;
import com.rti.dds.publication.DataWriterListener;
import com.rti.dds.topic.TypeSupportImpl;
// ===========================================================================
public class HelloWorldDataWriter extends DataWriterImpl {
    // -----------------------------------------------------------------------
    // Public Methods
    // -----------------------------------------------------------------------
    public InstanceHandle_t register_instance(HelloWorld instance_data) {
        return register_instance_untyped(instance_data);
    }
    public InstanceHandle_t register_instance_w_timestamp(HelloWorld instance_data,
    Time_t source_timestamp) {
        return register_instance_w_timestamp_untyped(
            instance_data, source_timestamp);
    }
    public InstanceHandle_t register_instance_w_params(HelloWorld instance_data,
    WriteParams_t params) {
        return register_instance_w_params_untyped(
            instance_data, params);
    }
    public void unregister_instance(HelloWorld instance_data,
    InstanceHandle_t handle) {
        unregister_instance_untyped(instance_data, handle);
    }
    public void unregister_instance_w_timestamp(HelloWorld instance_data,
    InstanceHandle_t handle, Time_t source_timestamp) {
        unregister_instance_w_timestamp_untyped(
            instance_data, handle, source_timestamp);
    }
    public void unregister_instance_w_params(HelloWorld instance_data,
    WriteParams_t params) {
        unregister_instance_w_params_untyped(
            instance_data, params);
    }
    public void write(HelloWorld instance_data, InstanceHandle_t handle) {
        write_untyped(instance_data, handle);
    }
    public void write_w_timestamp(HelloWorld instance_data,
    InstanceHandle_t handle, Time_t source_timestamp) {
        write_w_timestamp_untyped(instance_data, handle, source_timestamp);
    }
    public void write_w_params(HelloWorld instance_data,
    WriteParams_t params) {
        write_w_params_untyped(instance_data, params);
    }
    public void dispose(HelloWorld instance_data, InstanceHandle_t instance_handle){
        dispose_untyped(instance_data, instance_handle);
    }
    public void dispose_w_timestamp(HelloWorld instance_data,
    InstanceHandle_t instance_handle, Time_t source_timestamp) {
        dispose_w_timestamp_untyped(
            instance_data, instance_handle, source_timestamp);
    }
    public void dispose_w_params(HelloWorld instance_data,
    WriteParams_t params) {
        dispose_w_params_untyped(instance_data, params);
    }
    public void get_key_value(HelloWorld key_holder, InstanceHandle_t handle) {
        get_key_value_untyped(key_holder, handle);
    }
    public InstanceHandle_t lookup_instance(HelloWorld key_holder) {
        return lookup_instance_untyped(key_holder);
    }
```

```
    // ---------------------------------------------------------------------
    // Package Methods
    // ---------------------------------------------------------------------
    // --- Constructors: -------------------------------------------------
    /*package*/ HelloWorldDataWriter(long native_writer, DataWriterListener listener,
    int mask, TypeSupportImpl type) {
        super(native_writer, listener, mask, type);
    }
}
```

5.23 HelloWorldPublisher.java

[$(NDDSHOME)/example/JAVA/helloWorldPersistence/HelloWorldPublisher.java]

```
/* HelloWorldPublisher.java

   A publication of data of type HelloWorld

   This file is derived from code automatically generated by the rtiddsgen
   command:

   rtiddsgen -language java -example <arch> HelloWorld.idl

modification history
------------ -------
*/
import java.net.InetAddress;
import java.net.UnknownHostException;
import java.util.Arrays;
import com.rti.dds.domain.*;
import com.rti.dds.infrastructure.*;
import com.rti.dds.publication.*;
import com.rti.dds.topic.*;
import com.rti.ndds.config.*;
// ===========================================================================
public class HelloWorldPublisher {
    // ---------------------------------------------------------------------
    // Public Methods
    // ---------------------------------------------------------------------

    public static void main(String[] args) {
        // --- Get domain ID --- //
        int domainId = 0;
        if (args.length >= 1) {
            domainId = Integer.valueOf(args[0]).intValue();
        }
        // -- Get max loop count; 0 means infinite loop --- //
        int sampleCount = 0;
        if (args.length >= 2) {
            sampleCount = Integer.valueOf(args[1]).intValue();
        }

        /* Uncomment this to turn on additional logging
        Logger.get_instance().set_verbosity_by_category(
            LogCategory.NDDS_CONFIG_LOG_CATEGORY_API,
            LogVerbosity.NDDS_CONFIG_LOG_VERBOSITY_STATUS_ALL);
        */

        // --- Run --- //
        publisherMain(domainId, sampleCount);
    }



    // ---------------------------------------------------------------------
    // Private Methods
    // ---------------------------------------------------------------------

    // --- Constructors: -------------------------------------------------

    private HelloWorldPublisher() {
        super();
    }


    // ---------------------------------------------------------------------
```

```
    private static void publisherMain(int domainId, int sampleCount) {
        DomainParticipant participant = null;
        Publisher publisher = null;
        Topic topic = null;
        HelloWorldDataWriter writer = null;
        try {
            // --- Create participant --- //
            /* To create participant with default QoS,
               use DomainParticipantFactory.DomainParticipantFactory.
               participant.get_default_publisher_qos() */
            participant = DomainParticipantFactory.TheParticipantFactory.
                create_participant(
                    domainId, DomainParticipantFactory.PARTICIPANT_QOS_DEFAULT,
                    null /* listener */, StatusKind.STATUS_MASK_NONE);
            // --- Create publisher --- //
            /* To customize publisher QoS, use
               participant.get_default_publisher_qos() */
            publisher = participant.create_publisher(
                DomainParticipant.PUBLISHER_QOS_DEFAULT, null /* listener */,
                StatusKind.STATUS_MASK_NONE);

            // --- Create topic --- //
            /* Register type before creating topic */
            String typeName = HelloWorldTypeSupport.get_type_name();
            HelloWorldTypeSupport.register_type(participant, typeName);
            /* To customize topic QoS, use
               participant.get_default_topic_qos() */
            topic = participant.create_topic(
                "Example HelloWorld",
                typeName, DomainParticipant.TOPIC_QOS_DEFAULT,
                null /* listener */, StatusKind.STATUS_MASK_NONE);

            // --- Create writer --- //
            /* To customize data writer QoS, use
               publisher.get_default_datawriter_qos() */
            writer = (HelloWorldDataWriter)
                publisher.create_datawriter(
                    topic, Publisher.DATAWRITER_QOS_DEFAULT,
                    null /* listener */, StatusKind.STATUS_MASK_NONE);

            // --- Write --- //
            /* Create data sample for writing */
            HelloWorld instance = new HelloWorld();
            InstanceHandle_t instance_handle = InstanceHandle_t.HANDLE_NIL;
            /* For data type that has key, if the same instance is going to be
               written multiple times, initialize the key here
               and register the keyed instance prior to writing */
            //instance_handle = writer.register_instance(instance);
            final long sendPeriodMillis = 4 * 1000; // 4 seconds
            for (int count = 0;
                 (sampleCount == 0) || (count < sampleCount);
                 ++count) {
                System.out.println("Writing HelloWorld, count " + count);
                /* Modify the instance to be written here */
                instance.data = count;

                /* Write data */
                writer.write(instance, instance_handle);
                try {
                    Thread.sleep(sendPeriodMillis);
                } catch (InterruptedException ix) {
                    System.err.println("INTERRUPTED");
                    break;
                }
            }
            //writer.unregister_instance(instance, instance_handle);
        } finally {
            // --- Shutdown --- //
            if (participant != null) {
                participant.delete_contained_entities();
                DomainParticipantFactory.TheParticipantFactory.
                    delete_participant(participant);
            }
            /* RTI Connext provides finalize_instance()
               method for people who want to release memory used by the
               participant factory singleton. Uncomment the following block of
               code for clean destruction of the participant factory
               singleton. */
            //DomainParticipantFactory.finalize_instance();
        }
```

```
    }
}
```

5.24 HelloWorldSubscriber.java

[$(NDDSHOME)/example/JAVA/helloWorldPersistence/HelloWorldSubscriber.java]1

```
/* HelloWorldSubscriber.java

   A publication of data of type HelloWorld

   This file is derived from code automatically generated by the rtiddsgen
   command:

   rtiddsgen -language java -example <arch> HelloWorld.idl

modification history
------------ -------
*/
import java.net.InetAddress;
import java.net.UnknownHostException;
import java.util.Arrays;
import com.rti.dds.domain.*;
import com.rti.dds.infrastructure.*;
import com.rti.dds.subscription.*;
import com.rti.dds.topic.*;
import com.rti.ndds.config.*;
// ===========================================================================
public class HelloWorldSubscriber {
    // --------------------------------------------------------------------
    // Public Methods
    // --------------------------------------------------------------------

    public static void main(String[] args) {
        // --- Get domain ID --- //
        int domainId = 0;
        if (args.length >= 1) {
            domainId = Integer.valueOf(args[0]).intValue();
        }

        // -- Get max loop count; 0 means infinite loop --- //
        int sampleCount = 0;
        if (args.length >= 2) {
            sampleCount = Integer.valueOf(args[1]).intValue();
        }


        /* Uncomment this to turn on additional logging
        Logger.get_instance().set_verbosity_by_category(
            LogCategory.NDDS_CONFIG_LOG_CATEGORY_API,
            LogVerbosity.NDDS_CONFIG_LOG_VERBOSITY_STATUS_ALL);
        */

        // --- Run --- //
        subscriberMain(domainId, sampleCount);
    }



    // --------------------------------------------------------------------
    // Private Methods
    // --------------------------------------------------------------------

    // --- Constructors: -------------------------------------------------

    private HelloWorldSubscriber() {
        super();
    }


    // --------------------------------------------------------------------

    private static void subscriberMain(int domainId, int sampleCount) {
        DomainParticipant participant = null;
        Subscriber subscriber = null;
        Topic topic = null;
```

```
        DataReaderListener listener = null;
        HelloWorldDataReader reader = null;
        try {
            // --- Create participant --- //
            /* To create participant with default QoS,
               use DomainParticipantFactory.DomainParticipantFactory.
               participant.get_default_publisher_qos() */
            participant = DomainParticipantFactory.TheParticipantFactory.
                create_participant(
                    domainId, DomainParticipantFactory.PARTICIPANT_QOS_DEFAULT,
                    null /* listener */, StatusKind.STATUS_MASK_NONE);
            // --- Create subscriber --- //
            /* To customize subscriber QoS, use
               participant.get_default_subscriber_qos() */
            subscriber = participant.create_subscriber(
                DomainParticipant.SUBSCRIBER_QOS_DEFAULT, null /* listener */,
                StatusKind.STATUS_MASK_NONE);
            // --- Create topic --- //

            /* Register type before creating topic */
            String typeName = HelloWorldTypeSupport.get_type_name();
            HelloWorldTypeSupport.register_type(participant, typeName);
            /* To customize topic QoS, use
               participant.get_default_topic_qos() */
            topic = participant.create_topic(
                "Example HelloWorld",
                typeName, DomainParticipant.TOPIC_QOS_DEFAULT,
                null /* listener */, StatusKind.STATUS_MASK_NONE);

            // --- Create reader --- //
            listener = new HelloWorldListener();
            reader = (HelloWorldDataReader)
                subscriber.create_datareader(
                    topic, Subscriber.DATAREADER_QOS_DEFAULT, listener,
                    StatusKind.STATUS_MASK_ALL);

            // --- Wait for data --- //
            final long receivePeriodSec = 4;
            for (int count = 0;
                 (sampleCount == 0) || (count < sampleCount);
                 ++count) {
                System.out.println("HelloWorld subscriber sleeping for "
                                   + receivePeriodSec + " sec...");
                try {
                    Thread.sleep(receivePeriodSec * 1000);  // in millisec
                } catch (InterruptedException ix) {
                    System.err.println("INTERRUPTED");
                    break;
                }
            }
        } finally {
            // --- Shutdown --- //

            if(participant != null) {
                participant.delete_contained_entities();
                DomainParticipantFactory.TheParticipantFactory.
                    delete_participant(participant);
            }
            /* RTI Connext provides finalize_instance()
               method for people who want to release memory used by the
               participant factory singleton. Uncomment the following block of
               code for clean destruction of the participant factory
               singleton. */
            //DomainParticipantFactory.finalize_instance();
        }
    }

    // -----------------------------------------------------------------------
    // Private Types
    // -----------------------------------------------------------------------

    // =======================================================================

    private static class HelloWorldListener extends DataReaderAdapter {

        HelloWorldSeq _dataSeq = new HelloWorldSeq();
        SampleInfoSeq _infoSeq = new SampleInfoSeq();
        public void on_data_available(DataReader reader) {
            HelloWorldDataReader HelloWorldReader =
                (HelloWorldDataReader)reader;
```

```
            try {
                HelloWorldReader.take(
                    _dataSeq, _infoSeq,
                    ResourceLimitsQosPolicy.LENGTH_UNLIMITED,
                    SampleStateKind.ANY_SAMPLE_STATE,
                    ViewStateKind.ANY_VIEW_STATE,
                    InstanceStateKind.ANY_INSTANCE_STATE);
                for(int i = 0; i < _dataSeq.size(); ++i) {
                    SampleInfo info = (SampleInfo)_infoSeq.get(i);
                    if (info.valid_data) {
                        System.out.println(
                            ((HelloWorld)_dataSeq.get(i)).toString("Received",0));
                    }
                }
            } catch (RETCODE_NO_DATA noData) {
                // No data to process
            } finally {
                HelloWorldReader.return_loan(_dataSeq, _infoSeq);
            }
        }
    }
}
```

5.25 HelloWorldSeq.java

[$(NDDSHOME)/example/JAVA/helloWorldPersistence/HelloWorldSeq.java]

```
/*
WARNING: THIS FILE IS AUTO-GENERATED. DO NOT MODIFY.

This file was generated from .idl
using RTI Code Generator (rtiddsgen) version 4.7.0.
The rtiddsgen tool is part of the RTI Connext DDS distribution.
For more information, type 'rtiddsgen -help' at a command shell
or consult the Code Generator User's Manual.
*/
import java.util.Collection;
import com.rti.dds.infrastructure.Copyable;
import com.rti.dds.util.Enum;
import com.rti.dds.util.Sequence;
import com.rti.dds.util.LoanableSequence;
public final class HelloWorldSeq extends LoanableSequence implements Copyable {
    private static final long serialVersionUID = -882078169L;
    // --------------------------------------------------------------------
    // Package Fields
    // --------------------------------------------------------------------
    /*package*/ transient Sequence _loanedInfoSequence = null;
    // --------------------------------------------------------------------
    // Public Fields
    // --------------------------------------------------------------------
    // --- Constructors: ------------------------------------------------
    public HelloWorldSeq() {
        super(HelloWorld.class);
    }
    public HelloWorldSeq (int initialMaximum) {
        super(HelloWorld.class, initialMaximum);
    }
    public HelloWorldSeq (Collection<?> elements) {
        super(HelloWorld.class, elements);
    }
    public  HelloWorld get(int index) {
        return (HelloWorld) super.get(index);
    }
    // --- From Copyable: -----------------------------------------------
    @Override
    public java.lang.Object copy_from(java.lang.Object src) {
        Sequence typedSrc = (Sequence) src;
        final int srcSize = typedSrc.size();
        final int origSize = size();
        // if this object's size is less than the source, ensure we have
        // enough room to store all of the objects
        if (getMaximum() < srcSize) {
            setMaximum(srcSize);
        }
        // trying to avoid clear() method here since it allocates memory
        // (an Iterator)
```

```
        // if the source object has fewer items than the current object,
        // remove from the end until the sizes are equal
        if (srcSize < origSize){
            removeRange(srcSize, origSize);
        }
        // copy the data from source into this (into positions that already
        // existed)
        for(int i = 0; (i < origSize) && (i < srcSize); i++){
            if (typedSrc.get(i) == null){
                set(i, null);
            } else {
                // check to see if our entry is null, if it is, a new instance has to be allocated
                if (get(i) == null){
                    set(i, HelloWorld.create());
                }
                set(i, ((Copyable) get(i)).copy_from(typedSrc.get(i)));
            }
        }
        // copy 'new' HelloWorld objects (beyond the original size of this object)
        for(int i = origSize; i < srcSize; i++){
            if (typedSrc.get(i) == null) {
                add(null);
            } else {
                // NOTE: we need to create a new object here to hold the copy
                add(HelloWorld.create());
                // we need to do a set here since enums aren't truely Copyable
                set(i, ((Copyable) get(i)).copy_from(typedSrc.get(i)));
            }
        }
        return this;
    }
}
```

5.26 HelloWorldTypeCode.java

[$(NDDSHOME)/example/JAVA/helloWorldPersistence/HelloWorldTypeCode.java]

```
/*
WARNING: THIS FILE IS AUTO-GENERATED. DO NOT MODIFY.

This file was generated from .idl
using RTI Code Generator (rtiddsgen) version 4.7.0.
The rtiddsgen tool is part of the RTI Connext DDS distribution.
For more information, type 'rtiddsgen -help' at a command shell
or consult the Code Generator User's Manual.
*/
import com.rti.dds.typecode.*;
public class  HelloWorldTypeCode {
    public static final TypeCode VALUE_WO_MEMBERS = getTypeCodeWOMembers();
    // We need a type code without member in case of recursion
    @SuppressWarnings("cast")
    public static TypeCode getTypeCodeWOMembers() {
        TypeCode tc = null;
        StructMember sm[]=new StructMember[0];
        Annotations annotation = new Annotations();
        annotation.allowed_data_representation_mask(5);
        tc =
       TypeCodeFactory.TheTypeCodeFactory.create_struct_tc("HelloWorld",ExtensibilityKind.EXTENSIBLE_EXTENSIBILITY,
        sm , annotation);        return tc;
    }
    public static final TypeCode VALUE = getTypeCode();
    // Depending on the type represented in the IDL, we may perform some redundant
    // casts, we are suppressing that warning
    @SuppressWarnings("cast")
    private static TypeCode getTypeCode() {
        TypeCode tc = null;
        int __i=0;
        StructMember sm[]=new StructMember[1];
        Annotations memberAnnotations = null;
        memberAnnotations = new Annotations();
        memberAnnotations.default_annotation(AnnotationParameterValue.ZERO_LONG);
        memberAnnotations.min_annotation(AnnotationParameterValue.MIN_LONG);
        memberAnnotations.max_annotation(AnnotationParameterValue.MAX_LONG);
        sm[__i] = new  StructMember("data", false, (short)-1,  false, TypeCode.TC_LONG, 0, false,
       memberAnnotations , false /* must_understand */);__i++;
        Annotations annotation = new Annotations();
```

```
        annotation.allowed_data_representation_mask(5);
        tc =
       TypeCodeFactory.TheTypeCodeFactory.create_struct_tc("HelloWorld",ExtensibilityKind.EXTENSIBLE_EXTENSIBILITY,
        sm , annotation);
        return tc;
    }
}
```

5.27 HelloWorldTypeSupport.java

[$(NDDSHOME)/example/JAVA/helloWorldPersistence/HelloWorldTypeSupport.java]

```
/*
WARNING: THIS FILE IS AUTO-GENERATED. DO NOT MODIFY.

This file was generated from .idl
using RTI Code Generator (rtiddsgen) version 4.7.0.
The rtiddsgen tool is part of the RTI Connext DDS distribution.
For more information, type 'rtiddsgen -help' at a command shell
or consult the Code Generator User's Manual.
*/
import com.rti.dds.cdr.CdrEncapsulation;
import com.rti.dds.cdr.CdrInputStream;
import com.rti.dds.cdr.CdrOutputStream;
import com.rti.dds.cdr.CdrPrimitiveType;
import com.rti.dds.cdr.CdrBuffer;
import com.rti.dds.cdr.CdrHeader;
import com.rti.dds.dynamicdata.DynamicData;
import com.rti.dds.cdr.IllegalCdrStateException;
import com.rti.dds.publication.DataWriter;
import com.rti.dds.publication.DataWriterListener;
import com.rti.dds.subscription.DataReader;
import com.rti.dds.subscription.DataReaderListener;
import com.rti.dds.topic.DefaultEndpointData;
import com.rti.dds.topic.TypeSupportImpl;
import com.rti.dds.topic.TypeSupportType;
import com.rti.dds.infrastructure.*;
import com.rti.dds.infrastructure.RETCODE_ERROR;
import com.rti.dds.topic.PrintFormatProperty;
import com.rti.dds.topic.PrintFormatKind;
import com.rti.dds.typecode.TypeCode;
import com.rti.dds.typecode.ExtensibilityKind;
import com.rti.dds.topic.TypeSupportParticipantInfo;
import com.rti.dds.topic.TypeSupportEndpointInfo;
import com.rti.dds.domain.DomainParticipant;
public class HelloWorldTypeSupport extends TypeSupportImpl {
    // ---------------------------------------------------------------------
    // Private Fields
    // ---------------------------------------------------------------------
    private static final java.lang.String TYPE_NAME = "HelloWorld";
    private static final char[] PLUGIN_VERSION = {2, 0, 0, 0};
    private static final HelloWorldTypeSupport _singleton
    = new HelloWorldTypeSupport();
    // ---------------------------------------------------------------------
    // Public Methods
    // ---------------------------------------------------------------------
    // --- External methods: ----------------------------------------------
    /* The methods in this section are for use by users of RTI Connext
    */
    public static java.lang.String get_type_name() {
        return _singleton.get_type_nameI();
    }
    public static void register_type(DomainParticipant participant,
    java.lang.String type_name) {
        _singleton.register_typeI(participant, type_name);
    }
    public static void unregister_type(DomainParticipant participant,
    java.lang.String type_name) {
        _singleton.unregister_typeI(participant, type_name);
    }
    /* The methods in this section are for use by RTI Connext
    * itself and by the code generated by rtiddsgen for other types.
    * They should be used directly or modified only by advanced users and are
    * subject to change in future versions of RTI Connext.
    */
    public static HelloWorldTypeSupport get_instance() {
```

```
    return _singleton;
}
public static HelloWorldTypeSupport getInstance() {
    return get_instance();
}
public static TypeCode getTypeCode(){
    return HelloWorldTypeCode.VALUE;
}
@Override
public java.lang.Object create_data() {
    return HelloWorld.create();
}
@Override
public java.lang.Object _create_data_internal() {
    HelloWorld data = new HelloWorld();
    this._initialize_data_internal(data);
    return data;
}
public void _initialize_data_internal(HelloWorld data) {
}
public java.lang.Object _create_data_internal_seq(int size) {
    HelloWorldSeq data = new HelloWorldSeq(size);
    data.setObjectReuseMemoryManagement(size);
    for (int i = 0; i < size; ++i) {
        data.incrementSize();
        data.set(i, HelloWorldTypeSupport.get_instance()._create_data_internal());
    }
    data.clear();
    return data;
}
@Override
public void destroy_data(java.lang.Object data) {
    return;
}
@Override
public java.lang.Object create_key() {
    return new HelloWorld();
}
@Override
public void destroy_key(java.lang.Object key) {
    return;
}
@Override
public java.lang.Class<?> get_type() {
    return HelloWorld.class;
}
@Override
public java.lang.Object copy_data(java.lang.Object destination, java.lang.Object source) {
    HelloWorld typedDst = (HelloWorld) destination;
    HelloWorld typedSrc = (HelloWorld) source;
    return typedDst.copy_from(typedSrc);
}
@Override
public long get_serialized_sample_max_size(java.lang.Object endpoint_data,boolean
   include_encapsulation,short final_encapsulation_id,long currentAlignment) {
    CdrPrimitiveType _cdrPrimitiveType = CdrPrimitiveType.getInstance(final_encapsulation_id);
    short encapsulation_id = CdrEncapsulation.getEncapsulationFromFinal(
        final_encapsulation_id,
        ExtensibilityKind.EXTENSIBLE_EXTENSIBILITY);
    boolean xcdr1 = (encapsulation_id <= CdrEncapsulation.CDR_ENCAPSULATION_ID_PL_CDR_LE);
    DefaultEndpointData epd = (DefaultEndpointData) endpoint_data;
    if (epd == null) {
        /* Coverity reports that epd store a object that might not
        be used later. It is true that it might not be used,
        depending on the type complexity. This is intentional
        because it doesn't affect generated code and fixing it
        would make generated code more difficult to maintain */
        /* coverity[defect] */
        epd = new DefaultEndpointData();
    }
    long origAlignment = currentAlignment;
    long encapsulation_size = currentAlignment;
    if(include_encapsulation) {
        if (!CdrEncapsulation.isValidEncapsulationKind(encapsulation_id)) {
            throw new RETCODE_ERROR("Unsupported encapsulation");
        }
        encapsulation_size += _cdrPrimitiveType.getShortMaxSizeSerialized(encapsulation_size);
        encapsulation_size += _cdrPrimitiveType.getShortMaxSizeSerialized(encapsulation_size);
        encapsulation_size -= currentAlignment;
        currentAlignment = 0;
        origAlignment = 0;
```

```
            if(xcdr1){
                epd.setBaseAlignment(currentAlignment);
            }
        }
        if (!xcdr1) {
            currentAlignment += _cdrPrimitiveType.getIntMaxSizeSerialized(currentAlignment);
        }
        currentAlignment += _cdrPrimitiveType.getIntMaxSizeSerialized(epd.getAlignment(currentAlignment));
        if (include_encapsulation) {
            currentAlignment += encapsulation_size;
        }
        return currentAlignment - origAlignment;
    }
    @Override
    public long get_serialized_sample_min_size(java.lang.Object endpoint_data,boolean
       include_encapsulation,short final_encapsulation_id,long currentAlignment) {
        CdrPrimitiveType _cdrPrimitiveType = CdrPrimitiveType.getInstance(final_encapsulation_id);
        short encapsulation_id = CdrEncapsulation.getEncapsulationFromFinal(
            final_encapsulation_id,
            ExtensibilityKind.EXTENSIBLE_EXTENSIBILITY);
        boolean xcdr1 = (encapsulation_id <= CdrEncapsulation.CDR_ENCAPSULATION_ID_PL_CDR_LE);
        DefaultEndpointData epd = (DefaultEndpointData) endpoint_data;
        if (epd == null) {
            /* Coverity reports that epd store a object that might not
            be used later. It is true that it might not be used,
            depending on the type complexity. This is intentional
            because it doesn't affect generated code and fixing it
            would make generated code more difficult to maintain */
            /* coverity[defect] */
            epd = new DefaultEndpointData();
        }
        long origAlignment = currentAlignment;
        long encapsulation_size = currentAlignment;
        if(include_encapsulation) {
            if (!CdrEncapsulation.isValidEncapsulationKind(encapsulation_id)) {
                throw new RETCODE_ERROR("Unsupported encapsulation");
            }
            encapsulation_size += _cdrPrimitiveType.getShortMaxSizeSerialized(encapsulation_size);
            encapsulation_size += _cdrPrimitiveType.getShortMaxSizeSerialized(encapsulation_size);
            encapsulation_size -= currentAlignment;
            currentAlignment = 0;
            origAlignment = 0;
            if(xcdr1){
                epd.setBaseAlignment(currentAlignment);
            }
        }
        if (!xcdr1) {
            currentAlignment += _cdrPrimitiveType.getIntMaxSizeSerialized(currentAlignment);
        }
        currentAlignment += _cdrPrimitiveType.getIntMaxSizeSerialized(epd.getAlignment(currentAlignment));
        if (include_encapsulation) {
            currentAlignment += encapsulation_size;
        }
        return currentAlignment - origAlignment;
    }
    @Override
    public long get_serialized_sample_size(
        java.lang.Object endpoint_data, boolean include_encapsulation,
        short final_encapsulation_id, long currentAlignment,
        java.lang.Object sample)
    {
        CdrPrimitiveType _cdrPrimitiveType = CdrPrimitiveType.getInstance(final_encapsulation_id);
        short encapsulation_id = CdrEncapsulation.getEncapsulationFromFinal(
            final_encapsulation_id,
            ExtensibilityKind.EXTENSIBLE_EXTENSIBILITY);
        boolean xcdr1 = (encapsulation_id <= CdrEncapsulation.CDR_ENCAPSULATION_ID_PL_CDR_LE);
        HelloWorld typedSrc = (HelloWorld) sample;
        DefaultEndpointData epd = (DefaultEndpointData) endpoint_data;
        if (epd == null) {
            /* Coverity reports that epd store a object that might not
            be used later. It is true that it might not be used,
            depending on the type complexity. This is intentional
            because it doesn't affect generated code and fixing it
            would make generated code more difficult to maintain */
            /* coverity[defect] */
            epd = new DefaultEndpointData();
        }
        long origAlignment = currentAlignment;
        long encapsulation_size = currentAlignment;
        if(include_encapsulation) {
            if (!CdrEncapsulation.isValidEncapsulationKind(encapsulation_id)) {
```

```
            throw new RETCODE_ERROR("Unsupported encapsulation");
        }
        encapsulation_size += _cdrPrimitiveType.getShortMaxSizeSerialized(encapsulation_size);
        encapsulation_size += _cdrPrimitiveType.getShortMaxSizeSerialized(encapsulation_size);
        encapsulation_size -= currentAlignment;
        currentAlignment = 0;
        origAlignment = 0;
        if(xcdr1){
            epd.setBaseAlignment(currentAlignment);
        }
    }
    if (!xcdr1) {
        currentAlignment += _cdrPrimitiveType.getIntMaxSizeSerialized(currentAlignment);
    }
    currentAlignment += _cdrPrimitiveType.getIntMaxSizeSerialized(epd.getAlignment(currentAlignment));
    if (include_encapsulation) {
        currentAlignment += encapsulation_size;
    }
    return currentAlignment - origAlignment;
}
@Override
public long get_serialized_key_max_size(
    java.lang.Object endpoint_data,
    boolean include_encapsulation,
    short final_encapsulation_id,
    long currentAlignment)
{
    CdrPrimitiveType _cdrPrimitiveType = CdrPrimitiveType.getInstance(final_encapsulation_id);
    short encapsulation_id = CdrEncapsulation.getEncapsulationFromFinal(
        final_encapsulation_id,
        ExtensibilityKind.EXTENSIBLE_EXTENSIBILITY);
    boolean xcdr1 = (encapsulation_id <= CdrEncapsulation.CDR_ENCAPSULATION_ID_PL_CDR_LE);
    DefaultEndpointData epd = (DefaultEndpointData) endpoint_data;
    if (epd == null) {
        /* Coverity reports that epd store a object that might not
        be used later. It is true that it might not be used,
        depending on the type complexity. This is intentional
        because it doesn't affect generated code and fixing it
        would make generated code more difficult to maintain */
        /* coverity[defect] */
        epd = new DefaultEndpointData();
    }
    long origAlignment = currentAlignment;
    long encapsulation_size = currentAlignment;
    if(include_encapsulation) {
        if (!CdrEncapsulation.isValidEncapsulationKind(encapsulation_id)) {
            throw new RETCODE_ERROR("Unsupported encapsulation");
        }
        encapsulation_size += _cdrPrimitiveType.getShortMaxSizeSerialized(encapsulation_size);
        encapsulation_size += _cdrPrimitiveType.getShortMaxSizeSerialized(encapsulation_size);
        encapsulation_size -= currentAlignment;
        currentAlignment = 0;
        origAlignment = 0;
        if(xcdr1){
            epd.setBaseAlignment(currentAlignment);
        }
    }
    currentAlignment += get_serialized_sample_max_size(
        epd,false,final_encapsulation_id,currentAlignment);
    if (include_encapsulation) {
        currentAlignment += encapsulation_size;
    }
    return currentAlignment - origAlignment;
}
@Override
public long get_serialized_key_for_keyhash_max_size(
    java.lang.Object endpoint_data,
    boolean include_encapsulation,
    short final_encapsulation_id,
    long currentAlignment)
{
    CdrPrimitiveType _cdrPrimitiveType = CdrPrimitiveType.getInstance(final_encapsulation_id);
    short encapsulation_id = CdrEncapsulation.getEncapsulationFromFinal(
        final_encapsulation_id,
        ExtensibilityKind.EXTENSIBLE_EXTENSIBILITY);
    boolean xcdr1 = (encapsulation_id <= CdrEncapsulation.CDR_ENCAPSULATION_ID_PL_CDR_LE);
    DefaultEndpointData epd = (DefaultEndpointData) endpoint_data;
    if (epd == null) {
        /* Coverity reports that epd store a object that might not
        be used later. It is true that it might not be used,
        depending on the type complexity. This is intentional
```

```
            because it doesn't affect generated code and fixing it
            would make generated code more difficult to maintain */
            /* coverity[defect] */
            epd = new DefaultEndpointData();
        }
        if (xcdr1){
            return get_serialized_key_max_size(epd, include_encapsulation, final_encapsulation_id,
       currentAlignment) ;
        }
        long origAlignment = currentAlignment;
        long encapsulation_size = currentAlignment;
        if(include_encapsulation) {
            if (!CdrEncapsulation.isValidEncapsulationKind(encapsulation_id)) {
                throw new RETCODE_ERROR("Unsupported encapsulation");
            }
            encapsulation_size += _cdrPrimitiveType.getShortMaxSizeSerialized(encapsulation_size);
            encapsulation_size += _cdrPrimitiveType.getShortMaxSizeSerialized(encapsulation_size);
            encapsulation_size -= currentAlignment;
            currentAlignment = 0;
            origAlignment = 0;
        }
        currentAlignment += _cdrPrimitiveType.getIntMaxSizeSerialized(epd.getAlignment(currentAlignment));
        if (include_encapsulation) {
            currentAlignment += encapsulation_size;
        }
        return currentAlignment - origAlignment;
    }
    @Override
    public void serialize(java.lang.Object endpoint_data,java.lang.Object src,
    CdrOutputStream dst, boolean serialize_encapsulation, short final_encapsulation_id,
    boolean serialize_sample, java.lang.Object endpoint_plugin_qos) {
        int position = 0;
        int dheaderArrPosition = -1;
        int dheaderSeqPosition = -1;
        DefaultEndpointData epd = (DefaultEndpointData) endpoint_data;
        if (epd == null) {
            /* Coverity reports that epd store a object that might not
            be used later. It is true that it might not be used,
            depending on the type complexity. This is intentional
            because it doesn't affect generated code and fixing it
            would make generated code more difficult to maintain */
            /* coverity[defect] */
            epd = new DefaultEndpointData();
        }
        int dheaderPosition = -1;
        boolean inBaseClass_tmp = false;
        inBaseClass_tmp =  dst.inBaseClass;
        dst.inBaseClass = false;
        if(serialize_encapsulation) {
            dst.serializeAndSetCdrEncapsulation(final_encapsulation_id,
       ExtensibilityKind.EXTENSIBLE_EXTENSIBILITY);
            position = dst.resetAlignment();
        }
        if(serialize_sample) {
            boolean xcdr1 = (final_encapsulation_id <= CdrEncapsulation.CDR_ENCAPSULATION_ID_PL_CDR_LE)? true:
       false;
            if (!inBaseClass_tmp && !xcdr1) {
                dheaderPosition=dst.writeDHeader();
            }
            HelloWorld typedSrc = (HelloWorld) src;
            dst.writeInt(typedSrc.data);
            if (!xcdr1 && dheaderPosition != -1) {
                dst.setDHeader(dheaderPosition);
                dheaderPosition = -1;
            }
        }
        if (serialize_encapsulation) {
            dst.restoreAlignment(position);
        }
    }
    public long serialize_to_cdr_buffer(
        byte[] buffer,
        long length,
        HelloWorld src)
    {
        return super.serialize_to_cdr_buffer(buffer,length,src);
    }
    public long serialize_to_cdr_buffer(
        byte[] buffer,
        long length,
        HelloWorld src,
```

```
    short representation)
{
    return super.serialize_to_cdr_buffer(
        buffer,
        length,
        src,
        representation);
}
@Override
public void serialize_key(
    java.lang.Object endpoint_data,
    java.lang.Object src,
    CdrOutputStream dst,
    boolean serialize_encapsulation,
    short final_encapsulation_id,
    boolean serialize_key,
    java.lang.Object endpoint_plugin_qos)
{
    int dheaderArrPosition = -1;
    int dheaderSeqPosition = -1;
    int position = 0;
    DefaultEndpointData epd = (DefaultEndpointData) endpoint_data;
    if (epd == null) {
        /* Coverity reports that epd store a object that might not
        be used later. It is true that it might not be used,
        depending on the type complexity. This is intentional
        because it doesn't affect generated code and fixing it
        would make generated code more difficult to maintain */
        /* coverity[defect] */
        epd = new DefaultEndpointData();
    }
    boolean inBaseClass_tmp = false;
    inBaseClass_tmp =  dst.inBaseClass;
    dst.inBaseClass = false;
    if (serialize_encapsulation) {
        dst.serializeAndSetCdrEncapsulation(final_encapsulation_id,
   ExtensibilityKind.EXTENSIBLE_EXTENSIBILITY);
        position = dst.resetAlignment();
    } else {
        dst.setEncapsulationKind(final_encapsulation_id);
    }
    if (serialize_key) {
        boolean xcdr1 = (final_encapsulation_id <= CdrEncapsulation.CDR_ENCAPSULATION_ID_PL_CDR_LE)? true:
   false;
        HelloWorld typedSrc = (HelloWorld) src;
        dst.inBaseClass = false;
        serialize(epd, src, dst, false, final_encapsulation_id, true, endpoint_plugin_qos);
    }
    if (serialize_encapsulation) {
        dst.restoreAlignment(position);
    }
}
@Override
public void serialize_key_for_keyhash(
    java.lang.Object endpoint_data,
    java.lang.Object src,
    CdrOutputStream dst,
    boolean serialize_encapsulation,
    short final_encapsulation_id,
    boolean serialize_key,
    java.lang.Object endpoint_plugin_qos)
{
    int position = 0;
    int dheaderArrPosition = -1;
    int dheaderSeqPosition = -1;
    CdrPrimitiveType _cdrPrimitiveType = CdrPrimitiveType.getInstance(final_encapsulation_id);
    short encapsulation_id = CdrEncapsulation.getEncapsulationFromFinal(
        final_encapsulation_id,
        ExtensibilityKind.EXTENSIBLE_EXTENSIBILITY);
    boolean xcdr1 = (encapsulation_id <= CdrEncapsulation.CDR_ENCAPSULATION_ID_PL_CDR_LE);
    DefaultEndpointData epd = (DefaultEndpointData) endpoint_data;
    if (epd == null) {
        /* Coverity reports that epd store a object that might not
        be used later. It is true that it might not be used,
        depending on the type complexity. This is intentional
        because it doesn't affect generated code and fixing it
        would make generated code more difficult to maintain */
        /* coverity[defect] */
        epd = new DefaultEndpointData();
    }
    if (xcdr1){
```

```
            serialize_key (
                epd,
                src,
                dst,
                serialize_encapsulation,
                final_encapsulation_id,
                serialize_key,
                endpoint_plugin_qos);
        } else {
            if (serialize_encapsulation) {
                dst.serializeAndSetCdrEncapsulation(encapsulation_id);
                position = dst.resetAlignment();
            } else {
                /* We do this to prepare the stream to serialize using xcdr2 if needed
                * as in md5Stream we pass serialize_encapsulation ton false.
                */
                dst.setEncapsulationKind(final_encapsulation_id);
            }
            if (serialize_key) {
                HelloWorld typedSrc = (HelloWorld) src;
                dst.inBaseClass = false;
                dst.writeInt(typedSrc.data);
            }
            if (serialize_encapsulation) {
                dst.restoreAlignment(position);
            }
        }
    }
    // Depending on the type represented in the IDL, we may perform some redundant
    // casts, we are suppressing that warning
    @SuppressWarnings("cast")
    @Override
    public java.lang.Object deserialize_sample(
        java.lang.Object endpoint_data,
        java.lang.Object dst,
        CdrInputStream src, boolean deserialize_encapsulation,
        boolean deserialize_sample,
        java.lang.Object endpoint_plugin_qos)
    {
        int position = 0;
        int tmpPosition = 0, tmpSize = 0;
        long tmpLength = 0;
        CdrBuffer buffer = null;
        DefaultEndpointData epd = (DefaultEndpointData) endpoint_data;
        if (epd == null) {
            /* Coverity reports that epd store a object that might not
            be used later. It is true that it might not be used,
            depending on the type complexity. This is intentional
            because it doesn't affect generated code and fixing it
            would make generated code more difficult to maintain */
            /* coverity[defect] */
            epd = new DefaultEndpointData();
        }
        boolean inBaseClass_tmp = false;
        inBaseClass_tmp =  src.inBaseClass;
        src.inBaseClass = false;
        if(deserialize_encapsulation) {
            src.deserializeAndSetCdrEncapsulation();
            position = src.resetAlignment();
        }
        if(deserialize_sample) {
            short encapsulation_id = src.getEncapsulationKind();
            boolean xcdr1 = (encapsulation_id <= CdrEncapsulation.CDR_ENCAPSULATION_ID_PL_CDR_LE)? true: false;
            if(!xcdr1){
                buffer = src.getBuffer();
            }
            HelloWorld typedDst = (HelloWorld) dst;
            typedDst.clear();
            int DHtmpPosition = 0;
            int DHtmpSize = 0;
            long DHtmpLength = 0;
            if (!xcdr1 && !inBaseClass_tmp) {
                DHtmpLength = src.readInt();
                DHtmpPosition = buffer.currentPosition();
                DHtmpSize = buffer.getDesBufferSize();
                buffer.setDesBufferSize((int)(DHtmpPosition + DHtmpLength));
            }
            try{
                typedDst.data = src.readInt();
            } catch (IllegalCdrStateException stateEx) {
                if (src.available() >= CdrEncapsulation.CDR_ENCAPSULATION_PARAMETER_ID_ALIGNMENT) {
```

```
                throw new RETCODE_ERROR("Error deserializing sample! Remainder: " + src.available() + "\n" +
                "Exception caused by: " + stateEx.getMessage());
            }
        } catch (java.lang.Exception ex) {
            throw new RETCODE_ERROR(ex.getMessage());
        }
        if (!xcdr1 && !inBaseClass_tmp ) {
            buffer.restore(DHtmpSize, (int) (DHtmpPosition + DHtmpLength));
        }
    }
    if (deserialize_encapsulation) {
        src.restoreAlignment(position);
    }
    return dst;
}
public void deserialize_from_cdr_buffer(
    HelloWorld dst,
    byte[] buffer,
    long length)
{
    super.deserialize_from_cdr_buffer(dst,buffer,length);
}
public java.lang.String data_to_string(
    HelloWorld sample,
    PrintFormatProperty property)
{
    return super.data_to_string(sample, property);
}
public java.lang.String data_to_string(
    HelloWorld sample)
{
    return super.data_to_string(sample);
}
@java.lang.Override
public HelloWorld data_from_string(
    java.lang.String string,
    PrintFormatKind kind)
{
    return (HelloWorld) super.data_from_string(string, kind);
}
@java.lang.Override
public HelloWorld data_from_string(
    java.lang.String string)
{
    return (HelloWorld) super.data_from_string(string);
}
@java.lang.Override
public HelloWorld data_from_dynamicdata(
    DynamicData dynamicData)
{
    return (HelloWorld) super.data_from_dynamicdata(dynamicData);
}
@Override
public java.lang.Object deserialize_key_sample(
    java.lang.Object endpoint_data,
    java.lang.Object dst,
    CdrInputStream src,
    boolean deserialize_encapsulation,
    boolean deserialize_key,
    java.lang.Object endpoint_plugin_qos)
{
    int position = 0;
    int tmpPosition = 0, tmpSize = 0;
    long tmpLength = 0;
    CdrBuffer buffer = null;
    DefaultEndpointData epd = (DefaultEndpointData) endpoint_data;
    if (epd == null) {
        /* Coverity reports that epd store a object that might not
        be used later. It is true that it might not be used,
        depending on the type complexity. This is intentional
        because it doesn't affect generated code and fixing it
        would make generated code more difficult to maintain */
        /* coverity[defect] */
        epd = new DefaultEndpointData();
    }
    boolean inBaseClass_tmp = false;
    inBaseClass_tmp =  src.inBaseClass;
    src.inBaseClass = false;
    if(deserialize_encapsulation) {
        src.deserializeAndSetCdrEncapsulation();
        position = src.resetAlignment();
```

```
        }
        if(deserialize_key) {
            short encapsulation_id = src.getEncapsulationKind();
            boolean xcdr1 = (encapsulation_id <= CdrEncapsulation.CDR_ENCAPSULATION_ID_PL_CDR_LE)? true: false;
            if(!xcdr1){
                buffer = src.getBuffer();
            }
            HelloWorld typedDst = (HelloWorld) dst;
            deserialize_sample(epd, dst, src, false, true, endpoint_plugin_qos);
        }
        if (deserialize_encapsulation) {
            src.restoreAlignment(position);
        }
        return dst;
    }
    @Override
    public void skip(java.lang.Object endpoint_data,
    CdrInputStream src,
    boolean skip_encapsulation,
    boolean skip_sample,
    java.lang.Object endpoint_plugin_qos)
    {
        int position = 0;
        int tmpPosition = 0, tmpSize = 0;
        long tmpLength = 0;
        CdrBuffer buffer = null;
        DefaultEndpointData epd = (DefaultEndpointData) endpoint_data;
        if (epd == null) {
            /* Coverity reports that epd store a object that might not
            be used later. It is true that it might not be used,
            depending on the type complexity. This is intentional
            because it doesn't affect generated code and fixing it
            would make generated code more difficult to maintain */
            /* coverity[defect] */
            epd = new DefaultEndpointData();
        }
        boolean inBaseClass_tmp = false;
        inBaseClass_tmp =  src.inBaseClass;
        src.inBaseClass = false;
        if (skip_encapsulation) {
            src.skipEncapsulation();
            position = src.resetAlignment();
        }
        if (skip_sample) {
            short encapsulation_id = src.getEncapsulationKind();
            boolean xcdr1 = (encapsulation_id <= CdrEncapsulation.CDR_ENCAPSULATION_ID_PL_CDR_LE)? true: false;
            if(!xcdr1){
                buffer = src.getBuffer();
            }
            int DHtmpPosition = 0;
            long DHtmpLength = 0;
            if (!xcdr1 && !inBaseClass_tmp) {
                DHtmpLength = src.readInt();
                DHtmpPosition = buffer.currentPosition();
                buffer.setCurrentPosition((int) (DHtmpPosition + DHtmpLength));
                if (skip_encapsulation) {
                    src.restoreAlignment(position);
                }
                return;
            }
            try {
                src.skipInt();
            } catch (IllegalCdrStateException stateEx) {
                if (src.available() >=
                CdrEncapsulation.CDR_ENCAPSULATION_PARAMETER_ID_ALIGNMENT) {
                    throw new IllegalCdrStateException(
                        "Error skipping sample! Remainder:" + src.available()
                        + "\nException caused by: " + stateEx.getMessage());
                }
            }
        }
        if (skip_encapsulation) {
            src.restoreAlignment(position);
        }
    }
    @Override
    public java.lang.Object serialized_sample_to_key(
        java.lang.Object endpoint_data,
        java.lang.Object sample,
        CdrInputStream src,
        boolean deserialize_encapsulation,
```

```
        boolean deserialize_key,
        java.lang.Object endpoint_plugin_qos)
    {
        int position = 0;
        int tmpPosition = 0, tmpSize = 0;
        long tmpLength = 0;
        CdrBuffer buffer = null;
        DefaultEndpointData epd = (DefaultEndpointData) endpoint_data;
        if (epd == null) {
            /* Coverity reports that epd store a object that might not
            be used later. It is true that it might not be used,
            depending on the type complexity. This is intentional
            because it doesn't affect generated code and fixing it
            would make generated code more difficult to maintain */
            /* coverity[defect] */
            epd = new DefaultEndpointData();
        }
        boolean inBaseClass_tmp = false;
        inBaseClass_tmp =  src.inBaseClass;
        src.inBaseClass = false;
        if(deserialize_encapsulation) {
            src.deserializeAndSetCdrEncapsulation();
            position = src.resetAlignment();
        }
        if (deserialize_key) {
            short encapsulation_id = src.getEncapsulationKind();
            boolean xcdr1 = (encapsulation_id <= CdrEncapsulation.CDR_ENCAPSULATION_ID_PL_CDR_LE)? true: false;
            if(!xcdr1){
                buffer = src.getBuffer();
            }
            HelloWorld typedDst = (HelloWorld) sample;
            deserialize_sample(
                epd, sample, src, false,
                true, endpoint_plugin_qos);
        }
        if (deserialize_encapsulation) {
            src.restoreAlignment(position);
        }
        return sample;
    }
    // --------------------------------------------------------------------
    // Callbacks
    // --------------------------------------------------------------------
    @Override
    public Object on_participant_attached(java.lang.Object registration_data,
    TypeSupportParticipantInfo participant_info,
    boolean top_level_registration,
    java.lang.Object container_plugin_context,
    TypeCode type_code) {
        participant_info.xtypes_compliance_mask.set(com.rti.ndds.config.Compliance.get_xtypes_mask().get());
        return super.on_participant_attached(
            registration_data, participant_info, top_level_registration,
            container_plugin_context, type_code);
    }
    @Override
    public void on_participant_detached(java.lang.Object participant_data) {
        super.on_participant_detached(participant_data);
    }
    @Override
    public java.lang.Object on_endpoint_attached(java.lang.Object participantData,
    TypeSupportEndpointInfo endpoint_info,
    boolean top_level_registration,
    java.lang.Object container_plugin_context) {
        return super.on_endpoint_attached(
            participantData,  endpoint_info,
            top_level_registration, container_plugin_context);
    }
    @Override
    public void on_endpoint_detached(java.lang.Object endpoint_data) {
        super.on_endpoint_detached(endpoint_data);
    }
    // --------------------------------------------------------------------
    // Protected Methods
    // --------------------------------------------------------------------
    @Override
    protected DataWriter create_datawriter(long native_writer,
    DataWriterListener listener,
    int mask) {
        return new HelloWorldDataWriter (native_writer, listener, mask, this);
    }
    @Override
```

```
    protected DataReader create_datareader(long native_reader,
    DataReaderListener listener,
    int mask) {
        return new HelloWorldDataReader(native_reader, listener, mask, this);
    }
    // -----------------------------------------------------------------------
    // Constructor
    // -----------------------------------------------------------------------
    protected HelloWorldTypeSupport() {
        /* If the user data type supports keys, then the second argument
        to the constructor below should be true.  Otherwise it should
        be false. */
        super(TYPE_NAME, false, HelloWorldTypeCode.VALUE, HelloWorld.class,TypeSupportType.TST_STRUCT,
       PLUGIN_VERSION);
    }
    protected HelloWorldTypeSupport (boolean enableKeySupport) {
        super(TYPE_NAME, enableKeySupport, HelloWorldTypeCode.VALUE,
       HelloWorld.class,TypeSupportType.TST_STRUCT, PLUGIN_VERSION);
    }
}
```

Chapter 6

Data Structure Documentation

6.1 RTI_PersistenceService Struct Reference

RTI Persistence Service.

6.1.1 Detailed Description

RTI Persistence Service.

6.2 RTI_PersistenceServiceProperty Struct Reference

Configuration of RTI Persistence Service.

Data Fields

- char * **cfg_file**

  Path to RTI Persistence Service configuration file.
- const char ** **cfg_strings**

  XML configuration represented as strings.
- unsigned int **cfg_strings_count**

  *Size of the array **cfg_strings** (p. 94).*
- const char * **cfg_name**

  Configuration name. This parameter is required and it is used to find a <persistence_service> matching tag in the configuration file.
- char * **application_name**

  Assigns a name to the execution of the RTI Persistence Service.
- DDS_Boolean **identify_execution**

  Set this to true to append the host name and process ID to the RTI Persistence Service application name.

- int **domain_id**

  DDS domain ID for the DomainParticipants created by the service.
- DDS_Boolean **enable_administration**

  Set this to true to enable remote administration or false to disable it.
- int **administration_domain_id**

  *If **enable_administration** (p. 96) is true, this is the domain ID to use for remote administration.*
- int **thread_stack_size**

  RTI Persistence Service thread stack size, which controls the stack size for the main service thread.
- char * **info_dir**

  The info directory of the running RTI Persistence Service.
- int **restore**

  Indicates whether or not RTI Persistence Service must restore its state from the persistent storage.
- int **service_verbosity**

  The verbosity of RTI Persistence Service.
- int **dds_verbosity**

  The verbosity of RTI Connext Core Libraries.
- char * **license_file**

  Path to RTI Persistence Service license file.
- DDS_Boolean **enable_database_locking**

  Enable database locking.
- DDS_Boolean **initialize_dp_factory**

  Initialize the DomainParticipantFactory.

6.2.1 Detailed Description

Configuration of RTI Persistence Service.

This structure must be initialized with **RTI_PersistenceServiceProperty_INITIALIZER** (p. 21)

6.2.2 Field Documentation

6.2.2.1 cfg_file

```
char* RTI_PersistenceServiceProperty::cfg_file
```

Path to RTI Persistence Service configuration file.

If not `NULL`, this file is loaded; otherwise, if `cfg_strings` is not `NULL`, those XML strings are loaded.

[default] NULL.

6.2.2.2 cfg_strings

```
const char** RTI_PersistenceServiceProperty::cfg_strings
```

XML configuration represented as strings.

An array of strings that altogether make up an XML document to configure RTI Persistence Service. This parameter is used only if **cfg_file** (p. 94) is NULL.

For example:

```
unsigned int MY_PERSISTENCE_SERVICE_CFG_SIZE = 2;
const char *MY_PERSISTENCE_SERVICE_CFG[MY_PERSISTENCE_SERVICE_CFG_SIZE] = {
     "<dds><persistence_service> \
         <participant name=\"defaultParticipant\"> \
             <persistence_group name="\"groupName\">",
             "... \
     </dds>"};
property.cfg_strings = MY_PERSISTENCE_SERVICE_CFG;
property.cfg_strings_count = MY_PERSISTENCE_SERVICE_CFG_SIZE;
```

The reason for using an array instead of one single string is to get around the limited size of literal strings in C. In general, if you create the XML string dynamically using one single string in the array, setting **cfg_strings_count** (p. 95) to 1 is enough:

```
property.cfg_strings = malloc(sizeof(char *));
property.cfg_strings[0] = "<dds><routing_service>...</dds>";
property.cfg_strings_count = 1;
```

If your target system doesn't support a file system, you can configure the service using XML strings.

[default] NULL.

6.2.2.3 cfg_strings_count

```
unsigned int RTI_PersistenceServiceProperty::cfg_strings_count
```

Size of the array **cfg_strings** (p. 94).

[default] 0.

6.2.2.4 cfg_name

```
const char* RTI_PersistenceServiceProperty::cfg_name
```

Configuration name. This parameter is required and it is used to find a <persistence_service> matching tag in the configuration file.

[default] NULL.

6.2.2.5 application_name

```
char* RTI_PersistenceServiceProperty::application_name
```

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

Remote commands will refer to the RTI Persistence Service using this name. In addition, the name of DomainParticipants created by RTI Persistence Service will be based on this name.

[default] The value of cfg_name if different than NULL. Otherwise, "RTI_Persistence_Service".

6.2.2.6 identify_execution

```
DDS_Boolean RTI_PersistenceServiceProperty::identify_execution
```

Set this to true to append the host name and process ID to the RTI Persistence Service application name.

Used to get unique names for remote administration.

[default] DDS_BOOLEAN_FALSE

6.2.2.7 domain_id

```
int RTI_PersistenceServiceProperty::domain_id
```

DDS domain ID for the DomainParticipants created by the service.

[default] -1 (Use the XML value)

6.2.2.8 enable_administration

```
DDS_Boolean RTI_PersistenceServiceProperty::enable_administration
```

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

[default] DDS_BOOLEAN_FALSE

6.2.2.9 administration_domain_id

```
int RTI_PersistenceServiceProperty::administration_domain_id
```

If **enable_administration** (p. 96) is true, this is the domain ID to use for remote administration.

Takes precedence over the XML configuration. If **enable_administration** (p. 96) is false, this value is not used even if remote administration is enabled in the XML configuration.

[default] 0

6.2.2.10 thread_stack_size

```
int RTI_PersistenceServiceProperty::thread_stack_size
```

RTI Persistence Service thread stack size, which controls the stack size for the main service thread.

[default] RTI_OSAPI_THREAD_STACK_SIZE_DEFAULT

6.2.2.11 info_dir

```
char* RTI_PersistenceServiceProperty::info_dir
```

The info directory of the running RTI Persistence Service.

The service writes a ps.pid file into this directory when it is started. When the service finalizes the file is deleted.

The ps.pid file is used to check if the last execution of RTI Persistence Service finished gracefully or there were errors.

[default] NULL (the file ps.pid will not be used)

6.2.2.12 restore

```
int RTI_PersistenceServiceProperty::restore
```

Indicates whether or not RTI Persistence Service must restore its state from the persistent storage.

- Set to 0 to not restore the state
- Set to 1 to restore the state
- Set to -1 to use the XML value for restore

[default] -1 Use the XML value

6.2.2.13 service_verbosity

```
int RTI_PersistenceServiceProperty::service_verbosity
```

The verbosity of RTI Persistence Service.

Values:

- **RTI_PERSISTENCE_SERVICE_LOG_VERBOSITY_SILENT** (p. 21)
- **RTI_PERSISTENCE_SERVICE_LOG_VERBOSITY_EXCEPTIONS** (p. 21)
- **RTI_PERSISTENCE_SERVICE_LOG_VERBOSITY_WARNINGS** (p. 21)
- **RTI_PERSISTENCE_SERVICE_LOG_VERBOSITY_INFO** (p. 21)

[default] RTI_PERSISTENCE_SERVICE_LOG_VERBOSITY_EXCEPTIONS

6.2.2.14 dds_verbosity

```
int RTI_PersistenceServiceProperty::dds_verbosity
```

The verbosity of RTI Connext Core Libraries.

Values:

- **RTI_PERSISTENCE_SERVICE_LOG_VERBOSITY_IGNORE** (p. 22)
- **RTI_PERSISTENCE_SERVICE_LOG_VERBOSITY_SILENT** (p. 21)
- **RTI_PERSISTENCE_SERVICE_LOG_VERBOSITY_EXCEPTIONS** (p. 21)
- **RTI_PERSISTENCE_SERVICE_LOG_VERBOSITY_WARNINGS** (p. 21)
- **RTI_PERSISTENCE_SERVICE_LOG_VERBOSITY_INFO** (p. 21)

[default] RTI_PERSISTENCE_SERVICE_LOG_VERBOSITY_EXCEPTIONS

6.2.2.15 license_file

```
char* RTI_PersistenceServiceProperty::license_file
```

Path to RTI Persistence Service license file.

If not `NULL`, this file is checked for a valid license; otherwise, default location will be used. This parameter is only used if your installation requires a license file.

[default] NULL.

6.2.2.16 enable_database_locking

```
DDS_Boolean RTI_PersistenceServiceProperty::enable_database_locking
```

Enable database locking.

RTI Persistence Service will lock the database by creating an entry in a specific table in the database.

This entry is to avoid multiple instances of RTI Persistence Service accessing the same database at the same time.

[default] DDS_BOOLEAN_TRUE

6.2.2.17 initialize_dp_factory

```
DDS_Boolean RTI_PersistenceServiceProperty::initialize_dp_factory
```

Initialize the DomainParticipantFactory.

This field allows enabling or disabling the initialization of the DomainParticipantFactory when creating a Persistence Service instance.

When set to DDS_BOOLEAN_TRUE, the Persistence Service will initialize the DomainParticipantFactory from XML if a profile has is_default_participant_factory_profile attribute set to true. This is particularly useful for enabling the use of features such as Monitoring Library 2.0 in Persistence Service without requiring additional programmatic setup.

If your application has already initialized the DomainParticipantFactory before creating the Persistence Service instance, you may want to set this parameter to DDS_BOOLEAN_FALSE to prevent overwriting any QoS settings previously configured.

[default] DDS_BOOLEAN_TRUE

Index