RTI CORBA Compatibility Kit

CORBA-DDS Example Using Java

Instructions

Version 5.0

Your systems. Working as one.

Technical Support
Real-Time Innovations, Inc.
232 E. Java Drive
Sunnyvale, CA 94089

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

CORBA-DDS Example Using JAVA

This document will guide you through the steps required to create a CORBA-DDS publisher and a CORBA-DDS subscriber based on an IDL file. Then it will show you how to adapt those steps to create your own application (the provided example).

The example illustrates how to use the CORBA compatibility kit for *RTI® Connext* (formerly *RTI Data Distribution Service*) to create applications that use CORBA and DDS and share a common set of types for both APIs.

The example includes:

- ❏ A CORBA server application.
- ❏ A *Connext* subscriber application.
- ❏ A combined CORBA client-DDS publisher application. This application uses CORBA and DDS to send text messages to the CORBA server and the *Connext* subscriber. Both receiving applications print the messages to the console.

For more information, please see the *RTI CORBA Compatibility Kit Installation Guide* (**<NDDSHOME>/doc/pdf/RTI_CORBA_Compatibility_Kit_InstallationGuide.pdf**).

1 Building the Example

The example is in **<NDDSHOME>/example/JAVA/corba**. It includes the following:

- ❏ **Message.idl**: The data types for the example.
- ❏ **MessageApp.java:** Messaging application class, which contains the main method. Depending on the parameters passed at the command line, the executable will run the *Connext* subscriber, the CORBA server, or the sender/publisher.
- ❏ **MessageDDSPublisher.java:** Class used by the sender to publish messages over DDS.
- ❏ **MessageDDSSubscriber.java:** *Connext* subscriber receiving the sender messages and prints them on the screen.
- ❏ **MessageReceiverImpl.java:** Class implementing the CORBA server interface based on the IDL interface MessageReceiver. When the method sendMessage is invoked the CORBA server will print the input message on the screen.
- ❏ **IDL Generated files:** These files are generated automatically by the JacORB IDL compiler and by *rtiddsgen*.

Follow these steps to compile the example application:

1. Set Up Your Environment (Section 1.1)
2. Generate the Source Code (Section 1.2)
3. Modify the Publisher Data (Section 1.3)
4. Compile the Publisher and Subscriber (Section 1.4)
5. Run the Example Publisher and Subscriber (Section 1.5)
6. Use the Generated Files to Build the CORBA Example Application (Section 1.6)

1.1 Set Up Your Environment

Define the following environment variables:

❏ **JAVA_HOME**=<*path to your JDK installation directory*>

❏ **JAC_ORB**=<*path to your JAC-ORB installation directory*>

❏ **NDDSHOME**=<*path to your Connext installation directory*>

Examples:

```
setenv JAVA_HOME /local/Java/jdk1.5
setenv JAC_ORB /local/JacORB-2.2.4
setenv NDDSHOME /local/rti/ndds.5.0.x
```

1.2 Generate the Source Code

To generate the source code, use JacORB's IDL code generator and *rtiddsgen* (*Connext*'s IDL code generator):

1. Copy the example's IDL file (**Message.idl)** into a new directory of your choice.
2. Generate the CORBA support files by entering the following in a command shell:

   ```
   $JAVA_HOME/bin/java -classpath \
   $JAC_ORB/lib/idl.jar:$JAC_ORB/lib/logkit-1.2.jar \
   org.jacorb.idl.parser -d . Message.idl
   ```

   The above command will generate the following CORBA support files:

   - **DateTime.java, DateTimeHelper.java, DateTimeHolder.java**
   - **Message.java, MessageHelper.java, MessageHolder.java**
   - **MessageReceiver.java, MessageReceiverHelper.java**
   - **MessageReceiverHolder.java**
   - **MessageReceiverPOA.java, MessageReceiverPOATie.java**
   - **MessageReceiverOperations.java, _MessageReceiverStub.java**
   - **MSG_MAX_LENGTH.java**
   - **USER_QOS_PROFILES.XML**

3. Generate the *Connext* support files:

   ```
   $NDDSHOME/scripts/rtiddsgen -typecode -language Java \
    -example <java arch> -corba Message.idl
   ```

   In the above commands, replace `<java arch>` with your target architecture name. To recall your target architecture name, look in your **${NDDSHOME}/lib** directory; an example is **sparcSol2.10jdk.**

The above command will generate the following *Connext* support files:

- **DateTimeSeq.java, DateTimeTypeCode.java, DateTimeTypeSupport.java**
- **MessageSeq.java, MessageDataReader.java, MessageDataWriter.java**
- **MessageTypeSupport.java, MessageTypeCode.java**
- **MesssagePublisher.java, MessageSubscriber.java**
- **makefile_Message_<*java arch*>**

1.3 Modify the Publisher Data

You will need to modify the data sent by the DDS Publisher before compiling and running the Publisher and Subscriber. You can do so by editing **MessagePublisher.java** (in the same directory you created in Step 1 on Page 2) as follows:

```
for (int count = 0;
     (sampleCount == 0) || (count < sampleCount);
     ++count) {
    System.out.println("Writing Message, count " + count);

    /* Modify the instance to be written here */
    instance.time = new DateTime();
    instance.msg = "Hello World! (" + count + ")";

    /* Write data */
    writer.write(instance, instance_handle);
    try {
            Thread.sleep(sendPeriodMillis);
    } catch (InterruptedException ix) {
        System.err.println("INTERRUPTED");
        break;
    }
}
```

1.4 Compile the Publisher and Subscriber

In the same directory that you created in Step 1 on Page 2, enter the following:

```
gmake -f makefile_Message_<java arch>
```

This will create the **.class** files in the same directory.

Note: gmake will fail if you do not have **javac** on your path.

Please refer to the *RTI Core Libraries and Utilities Getting Started Guide* for more on how to compile *rtiddsgen*-generated code.

1.5 Run the Example Publisher and Subscriber

You have now successfully created a publisher and a subscriber for the CORBA data types included in the IDL file, **Message.idl**.

Run the applications in two separate terminals:

1.
   ```
   gmake ARGS="<Domain_ID> <Number_of_Samples>" \
   -f makefile_Message_<java arch> MessagePublisher
   ```

2.
   ```
   gmake ARGS="<Domain_ID> <Number_of_Samples>" \
   -f makefile_Message_<java arch> MessageSubscriber
   ```

Note that your environment will need to be set up according to Section 1.1 in each terminal.

Please refer to the *RTI Core Libraries and Utilities Getting Started Guide* for more on how to run *rtiddsgen*-generated Publisher and Subscriber applications.

Example output from **Message_publisher**:

```
Writing Message, count 0
Writing Message, count 1
Writing Message, count 2
```

Example output from **Message_subscriber**:

```
Received:
    time :
        year: 0
        month: 0
        day: 0
        hour: 0
        minute: 0
        second: 0
    msg: Hello World! (0)
```

1.6 Use the Generated Files to Build the CORBA Example Application

This section describes how to use the code generated by JacORB's IDL generator and *rtiddsgen* to build a custom application:

1. The custom application code has already been provided for you under **/example/JAVA/corba/** in the following files:
   - **MessageDDSPublisher.java**
   - **MessageDDSSubscriber.java**
   - **MessageReceiverImpl.java**
   - **MessageApp.java**

   Copy these files into the directory that contains the files you generated in Section 1.2.
2. Edit **makefile_Message_<java arch>** as follows:
   a. Add the new java files to **JAVA_SOURCES**.
   b. Add the following rule to run **MessageApp**:

      Note: You must use a TAB before $(JAVA_PATH), not spaces.

      On UNIX-based systems:

```
MessageApp: ./MessageApp.class
      $(JAVA_PATH) -classpath ".:$(RTI_CLASSPATH)" \
-Dorg.omg.CORBA.ORBClass=org.jacorb.orb.ORB \
-Dorg.omg.CORBA.ORBSingletonClass=org.jacorb.orb.ORBSingleton \
MessageApp $(ARGS)
```

      On Windows systems:

```
MessageApp: ./MessageApp.class
      $(JAVA_PATH) -classpath ".;$(RTI_CLASSPATH)" \
-Dorg.omg.CORBA.ORBClass=org.jacorb.orb.ORB \
-Dorg.omg.CORBA.ORBSingletonClass=org.jacorb.orb.ORBSingleton \
MessageApp $(ARGS)
```

   c. Save the new makefile as **makefile_MessageApp**.
3. Compile your application:

```
gmake -f makefile_MessageApp
```

2 Running the Example

You will need three separate command shells.

1. In the first command shell, run the CORBA server:

   ```
   > gmake -f makefile_MessageApp MessageApp ARGS="-cr Message.ior"
   ```

   Message.ior is the name for a file that will be created by the CORBA server within the application to store its IOR (Interoperable Object Reference). The client will use this file in Step 3.

2. In the second terminal, run the *Connext* subscriber:

   ```
   > gmake -f makefile_MessageApp MessageApp ARGS="-nr <domainId>"
   ```

   The domain ID is required. Only applications using the same domain ID will communicate with each other.

3. In the third terminal, run the client (sender) application:

   ```
   > gmake -f makefile_MessageApp MessageApp \
        ARGS="-s Message.ior <domainId>"
   ```

   The domain ID is required. Use the same value that you used for the subscriber.

   The application will output a prompt as follows:

   ```
   sender>
   ```

4. Type some messages at the prompt; they will be sent to the CORBA server and to the *Connext* subscriber applications.