RTI Database Integration Service

Getting Started Guide

Version 5.3.1

Your systems. Working as one.

Printed in U.S.A. First printing.
February 2018.

Trademarks

Third Party Copyright Notices

Copy and Use Restrictions

Technical Support

Real-Time Innovations, Inc.
232 E. Java Drive
Sunnyvale, CA 94089
Phone: (408) 990-7444
Email: support@rti.com
Website: https://support.rti.com/

Available Documentation

The following documentation is available for *RTI® Database Integration Service* (formerly known as *RTI Real-Time Connect*):

- *Release Notes*, **RTI_Database_Integration_Service_ReleaseNotes.pdf**. This document provides an overview of the current release's features and lists changes since the previous release, system requirements, supported architectures, and compatibility with other products.
- *Getting Started Guide*, **RTI_Database_Integration_Service_GettingStarted.pdf**. This document provides installation instructions, a short 'Hello World' tutorial, and troubleshooting tips.
- *User's Manual*, **RTI_Database_Integration_Service_UsersManual.pdf**. This document starts with an overview of *Database Integration Service*'s basic concepts, terminology, and unique features. It then describes how to develop and implement applications that use *Database Integration Service*.

Additional Resources

- The *ODBC API Reference* from Microsoft is available from http://msdn.microsoft.com/en-us/library/ms714562(VS.85).aspx.
- The documentation for the Oracle TimesTen In-Memory Database can be found in the **doc/** directory of the Oracle TimesTen installation.
- Documentation for Oracle databases: http://www.oracle.com/technology/documentation/index.html.
- Documentation for MySQL: http://dev.mysql.com/doc/#manual.
- Documentation for PostgreSQL: https://www.postgresql.org/docs.

Contents

4 Database Integration Service for Oracle 11g on UNIX-based Systems

5 Database Integration Service for Oracle 11g on Windows Systems

6 Database Integration Service for TimesTen with Oracle In-Memory Database Cache

7 Database Integration Service for MySQL on UNIX-based Systems

8 Database Integration Service for MySQL on Windows Systems

9 Database Integration Service for Microsoft SQL Server

10 Database Integration Service for PostgreSQL on UNIX-Based Systems

Chapter 1 Welcome to RTI Database Integration Service

Welcome to *RTI® Database Integration Service*—a high-performance solution for integrating applications and data across real-time and enterprise systems from RTI.

Database Integration Service is the integration of two complementary technologies: data-centric publish-subscribe middleware and relational database management systems (RDBMS). This powerful integration allows your applications to uniformly access data from real-time/embedded and enterprise data sources via *RTI Connext® DDS*, or via database interfaces. Since both these technologies are data-centric and complementary, they can be combined to enable a new class of applications. In particular, *Connext DDS* can be used to produce a truly decentralized, distributed RDBMS, while RDBMS technology can be used to provide persistence for real-time data.

- **Intended Readers**

  This document is intended for system administrators and others who are responsible for performing installation and configuration tasks.

  This document guides you through the process of installing *Database Integration Service* and running three different scenarios:

- Storing Samples From Publishers
- Publishing Database Updates to Subscribers
- Database Table Replication. We only cover this scenario for Oracle® TimesTen. Database replication is also possible with MySQL™ and Oracle 11g.

The following platform configurations are covered. After reading this introduction, you can skip to the chapter that discusses the configuration that you will be using.

Note: *Database Integration Service* for Oracle and *Database Integration Service* for Oracle MySQL require that *Connext DDS* is already installed.

To use Oracle TimesTen, read either:

- Chapter 2: Database Integration Service for Oracle TimesTen on UNIX-Based Systems
- Chapter 3: Database Integration Service for Oracle TimesTen on Windows Systems
- Additionally, to use Oracle In-Memory Database Cache, read Chapter 6: Database Integration Service for TimesTen with Oracle In-Memory Database Cache

To use Oracle 11g, read either:

- Chapter 4: Database Integration Service for Oracle 11g on UNIX-based Systems
- Chapter 5: Database Integration Service for Oracle 11g on Windows Systems

To use MySQL, read either:

- Chapter 7: Database Integration Service for MySQL on UNIX-based Systems

- Chapter 8: Database Integration Service for MySQL on Windows Systems

To use Microsoft® SQL Server®, read:

- Chapter 9: Database Integration Service for Microsoft SQL Server

To use PostgreSQL®, read:

- Chapter 10: Database Integration Service for PostgreSQL on UNIX-Based Systems

1.1 Paths Mentioned in Documentation

The documentation refers to:

- **<NDDSHOME>**

  This refers to the installation directory for *Connext DDS*.

  The default installation paths are:

  - Mac® OS X systems:

    /Applications/rti_connext_dds-*version*

  - UNIX-based systems, non-*root* user:

    /home/*your user name*/rti_connext_dds-*version*

  - UNIX-based systems, *root* user:

    /opt/rti_connext_dds-*version*

  - Windows® systems, user without Administrator privileges:

    <*your home directory*>\rti_connext_dds-*version*

  - Windows systems, user with Administrator privileges:

    C:\Program Files\rti_connext_dds-*version* (for 64-bits machines) or
    C:\Program Files (x86)\rti_connext_dds-*version* (for 32-bit machines)

  You may also see $NDDSHOME or %NDDSHOME%, which refers to an environment variable set to the installation path.

  Wherever you see <NDDSHOME> used in a path, replace it with your installation path.

  Note for Windows Users: When using a command prompt to enter a command that includes the path **C:\Program Files** (or any directory name that has a space), enclose the path in quotation marks. For example:

  ```
  "C:\Program Files\rti_connext_dds-version\bin\rtiddsgen"
  ```

  or if you have defined the NDDSHOME environment variable:

  ```
  "%NDDSHOME%\bin\rtiddsgen"
  ```

- **RTI Workspace directory, rti_workspace**

  The RTI Workspace is where all configuration files for the applications and example files are located. All configuration files and examples are copied here the first time you run *RTI Launcher* or any script in **<NDDSHOME>/bin**. The default path to the RTI Workspace directory is:

  - Mac OS X systems:

    /Users/*your user name*/rti_workspace

- UNIX-based systems:

  /home/*your user name*/rti_workspace

- Windows systems:

  ***your Windows documents folder*\rti_workspace**

  Note: '*your Windows documents folder*' depends on your version of Windows.
  For example, on Windows 7, the folder is **C:\Users\your user name\Documents**; on Windows Server 2003, the folder is **C:\Documents and Settings\your user name\Documents**.

You can specify a different location for the **rti_workspace** directory. See the *RTI Connext DDS Core Libraries Getting Started Guide* for instructions.

- **<path to examples>**

  Examples are copied into your home directory the first time you run *RTI Launcher* or any script in **<NDDSHOME>/bin**. This document refers to the location of these examples as **<path to examples>.** Wherever you see <path to examples>, replace it with the appropriate path.

  By default, the examples are copied to **rti_workspace/*version*/examples**

  So the paths are:

  - Mac OS X systems:

    /Users/*your user name*/rti_workspace/*version*/examples

  - UNIX-based systems:

    /home/*your user name*/rti_workspace/*version*/examples

  - Windows systems:

    ***your Windows documents folder*\rti_workspace*version*\examples**

    Note: '*your Windows documents folder*' is described above.

  You can specify that you do not want the examples copied to the workspace. See the *RTI Connext DDS Core Libraries Getting Started Guide* for instructions.

Chapter 2 Database Integration Service for Oracle TimesTen on UNIX-Based Systems

2.1 Installation

1. Install *Database Integration Service* on top of *RTI Connext DDS*. There are two ways to install it, from *RTI Launcher* or from the command line.

   <NDDSHOME> is described in Paths Mentioned in Documentation (Section 1.1).

   To install from RTI Launcher:

   a. Start *RTI Launcher*:

   ```
   cd <NDDSHOME>/bin
   ./rtilauncher
   ```

   b. From the Utilities tab, click on **RTI Package Installer**.

   c. Use the + sign to add the **.rtipkg** file that you want to install.

   d. Click **Install**.

   To install from the command line:

   ```
   cd <NDDSHOME>/bin
   ./rtipkginstall <path to .rtipkg file>
   ```

2. Add the path to the Oracle TimesTen libraries to the beginning of **LD_LIBRARY_PATH**. For example:

   ```
   > setenv LD_LIBRARY_PATH /opt/TimesTen/tt1121/lib:$LD_LIBRARY_PATH
   ```

   or

   ```
   > export LD_LIBRARY_PATH=/opt/TimesTen/tt1121/lib:$LD_LIBRARY_PATH
   ```

3. Add the path to the Oracle TimesTen utilities to the beginning of **PATH**. For example:

   ```
   > setenv PATH /opt/TimesTen/tt1121/bin:$PATH
   ```

   or

   ```
   > export PATH=/opt/TimesTen/tt1121/bin:$PATH
   ```

2.2 Creating an Oracle TimesTen Database

Before you can use *Database Integration Service* for Oracle TimesTen, you need to get a TimesTen user account to an existing data store (database) from your database administrator. This section explains how to create a TimesTen database when you act as your own database and system administrator.

- Log in as the Oracle TimesTen administrator (the user that installed TimesTen):

```
> su - root
```

- **Create an ODBC INI file called .odbc.ini under the administrator's home directory. This file will contain the DSN (Data Source Name) that will provide access to the data store.**
- Insert the following lines in the file:

```
[ODBC Data Source]
Example=TimesTen Driver

[Example]
DRIVER=/opt/TimesTen/tt1121/lib/libtten.so
DataStore=/tmp/Example
DatabaseCharacterSet=AL32UTF8
```

Note: Make sure that **DRIVER** points to the valid location of the Oracle TimesTen ODBC driver on your system. You may want to choose a different **DataStore** path than the one shown above.

Note: If you want to use *Database Integration Service* for Oracle TimesTen with Oracle In-Memory Database Cache (IMDB) (see Chapter 6: Database Integration Service for TimesTen with Oracle In-Memory Database Cache), add the attribute **OracleNetServiceName** and be sure that the attribute **DatabaseCharacterSet** is set to the same character set that Oracle 11g uses. For example:

```
[Example]
DRIVER=/opt/TimesTen/tt1121/lib/libtten.so
DataStore=/tmp/Example
DatabaseCharacterSet=WE8ISO8859P1
OracleNetServiceName=orcl11g
```

You can determine the Oracle database character set by executing the following query on SQL*Plus as any user:

```
SQL> SELECT value FROM nls_database_parameters WHERE parameter='NLS_CHARACTER-
SET';
```

- **Create the new data store by executing the following command:**

```
> ttlsql -connStr "DSN=Example"
command> exit
```

2.3 Creating an Oracle TimesTen User Account

Starting with Oracle Timesten 11.2.1, database access control is mandatory.

If you are acting as your own database and system administrator, use the following steps to create an account for the user “Student”:

1. If Oracle TimesTen was not installed with world accessibility, the operating system user (for example, “oracle”) who is going to log in to the database as “Student” must be added to the Oracle Timesten user’s group. For example:

   ```
   > su - root
    > useradd -G <TimesTen users group> oracle
   ```

   For additional information on installation requirements for operating system group and file permissions, see the *Oracle TimesTen In-Memory Database Installation Guide*.

2. Log in as the Oracle TimesTen administrator (the user that installed TimesTen):

   ```
    > su - root
   ```

3. Connect to the Oracle TimesTen database where the user account must be created[1]:

   ```
    > ttlsql -connStr "DSN=Example"
   ```

4. Enter the following to create a user account with name “Student” and password “mypsswrd”:

   ```
   Command> create user Student identified by mypsswrd;
   Command> grant all privileges to Student;
   Command> exit
   ```

2.4 Creating a Data Source for Oracle TimesTen

Database Integration Service for Oracle TimesTen uses the Oracle TimesTen ODBC driver to access data sources (datastores). Usually these are the same data sources to which your applications connect. The connection information for each data source is stored in the **.odbc.ini** file. The stored information describes each data source in detail, specifying the driver name, a description, and any additional information the driver needs to connect to the data source.

To create the **.odbc.ini** file, follow these steps:

1. Create a new text file named **.odbc.ini** in your home directory. Alternatively, you can use the **ODBCINI** environment variable to specify the file location.
2. Insert these lines in the file:

   ```
   [ODBC Data Source]
   Example=TimesTen Driver

   [Example]
   DRIVER=/opt/TimesTen/tt1121/lib/libtten.so
   DataStore=/tmp/Example
   UID=Student
   PWD=mypsswrd
   DatabaseCharacterSet=AL32UTF8
   ```

1. If *ttlsql* is not on your path, it can be found in the **bin/** directory of your Oracle TimesTen installation.

Note: If you want to use *Database Integration Service* for Oracle TimesTen with Oracle In-Memory Database Cache (IMDB) (see Chapter 6: Database Integration Service for TimesTen with Oracle In-Memory Database Cache), you must add the attributes **OracleNetServiceName** and **OraclePWD** to the DSN. In addition, the attribute **DatabaseCharacterSet** must be set to the same character set that Oracle 11g uses. For example:

```
[Example]
DRIVER=/opt/TimesTen/tt1121/lib/libtten.so
DataStore=/tmp/Example
UID=Student
PWD=mypsswrd
OraclePWD=mypsswrd
DatabaseCharacterSet=WE8ISO8859P1
 OracleNetServiceName=orcl11g
```

3. Save your changes.

2.5 Creating a Configuration File for Database Integration Service

Database Integration Service reads its configuration information from an XML file. By default, *Database Integration Service* tries to load the configuration file, ***<Database Integration Service executable location>*/../../resource/xml/RTI_REAL_TIME_CONNECT.xml**. You can specify a different file with the command line option, **-cfgFile**.

The default file, **RTI_REAL_TIME_CONNECT.xml**, does not contain any valid database configuration information yet. For this example, we will edit this file as follows:

1. Look for the tag **<timesten_connection>**. Replace the tags **<dsn>**, **<user_name>**, and **<password>** as follows:

```
<timesten_connection>
  <dsn>Example</dsn>
  <user_name>Student</user_name>
  <password>mypsswrd</password>
</timesten_connection>
```

2. Save the file.

This configuration file instructs *Database Integration Service* to monitor the data source as specified by the "Example" DSN.

2.6 Starting Database Integration Service for Oracle TimesTen Daemon

Start *Database Integration Service* for Oracle TimesTen by executing the following command from **<NDDSHOME>/bin**.

<NDDSHOME> is described in Paths Mentioned in Documentation (Section 1.1).

```
> ./rtirtc_timesten -noDaemon -cfgName default
```

By default, *Database Integration Service* for Oracle TimesTen runs in the background as a daemon process. Specifying the **-noDaemon** option prevents that and the *Database Integration Service* Daemon starts as a regular process. Messages are sent to standard output.

You should see output similar to the following, indicating that the process is running.

```
>./rtirtc_timesten -noDaemon -cfgName default
RTI Real-Time Connect to TimesTen, Release 5.x.y: startup succeeded
```

Database Integration Service for Oracle TimesTen is now connected to the **Example** data source.

Note: Make sure that you have the Oracle TimesTen libraries in your **LD_LIBRARY_PATH**, see Step 3 of Installation (Section 2.1).

2.7 Storing Samples from Publishers

In this section, you will enable automatic capturing of data samples in an Oracle TimesTen data store. The first step is to create an IDL type definition. By using *RTI Code Generator's*[1] **-example** option, you will automatically create a Publisher of this IDL type.

1. Create a new text file called "**MyType.idl**" with the following contents:

```
struct MyType {
    short pkey; //@key
    char message[13];
};
```

   This IDL file specifies a data type that contains a message. Each instance is uniquely identified by the "**pkey**" field.

2. Now run the following command to compile the IDL type:

```
> rtiddsgen -language C -example <arch> MyType.idl
```

   For example:

```
> rtiddsgen -language C -example i86Linux2.6gcc4.1.1 MyType.idl
```

   This generates the **MyType**, **MyTypePlugin**, and **MyTypeSupport** files, as well as the **MyType_publisher** and **MyType_subscriber** example code.

3. The generated example will have a makefile named:

```
makefile_MyType_<arch>
```

   You may need to edit the makefile to specify the location of the compiler if it is not available on your path.

4. Edit **MyType_publisher.c**, and find the line containing the comment:

```
/* Modify the data to be written here */
```

   Insert the following lines immediately below this comment:

```
instance->pkey = count;
strcpy(instance->message, "Hello world!");
```

5. Save your changes and build the **MyType_publisher** and **MyType_subscriber** applications by executing:

```
> gmake -f makefile_MyType_<arch>
```

6. Start the **MyType_publisher** application so that it starts publishing data samples.

```
> objs/<arch>/MyType_publisher
```

   On the screen, you will see:

1. *RTI Code Generator* (*rtiddsgen*) is an IDL code generator distributed with *Connext DDS*. Please see the *RTI Code Generator User's Manual* for more information.

```
Writing MyType, count 0
Writing MyType, count 1
Writing MyType, count 2
...
```

The samples are not captured in the Oracle TimesTen data store yet. For this, you need to set up a subscription in *Database Integration Service* for Oracle.

Subscriptions are set up in the **RTIDDS_SUBSCRIPTIONS** configuration table that *Database Integration Service* for Oracle TimesTen created when it connected to the Oracle TimesTen data store. To insert entries into this table, you can use the Oracle TimesTen interactive SQL utility.

7. Start the TimesTen Interactive SQL utility from the command prompt:

```
> ttlsql -connStr "DSN=Example"
```

8. You can see that the **RTIDDS_SUBSCRIPTIONS** configuration table is still empty at this point by executing the following SQL command—*don't forget to type a semicolon ';' at the end of the line:*

```
Command> select * from RTIDDS_SUBSCRIPTIONS;
0 rows found.
```

9. To store the samples from the **MyType_publisher** application in a table named **Example**, insert a corresponding entry into the **RTIDDS_SUBSCRIPTIONS** table:

```
Command> insert into RTIDDS_SUBSCRIPTIONS values
('Student', 'Example', 0, 'Example MyType', 'MyType');
1 row inserted.
```

This entry directs *Database Integration Service* for Oracle TimesTen to create a user table named **Example** with owner **Student** and to start storing samples published with topic **Example MyType** and data type **MyType** in domain **0**.

10. If the **MyType_publisher** application is still running, you can execute the following SQL statement to view the contents of the table—otherwise, restart **MyType_publisher** as described above before executing this statement.

```
Command> select * from Example;
```

The output will look something like this:

```
...
< 134, Hello world!\000, 0, 1 >
< 135, Hello world!\000, 0, 1 >
< 136, Hello world!\000, 0, 1 >
85 rows found.
```

The actual number of rows found depends on when exactly the *Database Integration Service* Daemon started storing samples. If you execute the **select** statement repeatedly you will see the number of rows grow. This is because the **MyType_publisher** application writes a new instance every 4 seconds.

2.8 Publishing Database Updates to Subscribers

In this section, you will enable *Database Integration Service* for Oracle TimesTen to automatically publish changes made to a table in a data source. For this, you will again use the Oracle TimesTen Interactive SQL utility to change a record in the "**Example**" table. We assume that you have followed the instructions in Starting Database Integration Service for Oracle TimesTen Daemon (Section 2.6) to start a *Database*

Integration Service Daemon. In addition, we assume that you have followed the instructions in Steps 1 through 4 in Storing Samples from Publishers (Section 2.7) to create the IDL file and generate and compile the example code.

1. If you have not already done so, stop the **MyType_publisher** application by pressing "**Ctrl-c**" in the window where it is running.
2. Start the **MyType_subscriber** application from the command line:

   ```
   > objs/<arch>/MyType_subscriber
   MyType subscriber sleeping for 4 sec...
   MyType subscriber sleeping for 4 sec...
   ...
   ```

   You will see this message repeatedly, since nothing is being published.
3. For publishing changes to a data source, you need to configure the **RTIDDS_PUBLICATIONS** table. For publishing the changes to the **Example** table, execute the following SQL statement:

   ```
   Command> insert into RTIDDS_PUBLICATIONS values
           ('Student', 'Example', 0, 'Example MyType', 'MyType');
   ```

   This entry directs *Database Integration Service* for Oracle TimesTen to start publishing changes to table **Example** of owner **Student** as the topic **Example MyType** with type **MyType** in domain **0**.
4. Now change one of the previously captured samples in the **Example** table, for instance the last one:

   ```
   Command> update Example set message = 'Hello again!' where
            pkey in (select max(pkey) from Example);
   ```

   You will see that the **MyType_subscriber** application reports the update, for example:

   ```
   ...
   MyType subscriber sleeping for 4 sec...
   pkey: 748
      message:
      message[ 0]: 'H'
      message[ 1]: 'e'
      message[ 2]: 'l'
      message[ 3]: 'l'
      message[ 4]: 'o'
      message[ 5]: ' '
      message[ 6]: 'a'
      message[ 7]: 'g'
      message[ 8]: 'a'
      message[ 9]: 'i'
      message[10]: 'n'
      message[11]: '!'
      message[12]: ' '
   MyType subscriber sleeping for 4 sec...
   ...
   ```
5. You can also update all entries in the table:

   ```
   Command> update Example set message = 'Hello again!';
   ```

   Notice that the **MyType_subscriber** application receives all changes, possibly a large number.

2.9 Database Table Replication

This section explains how to use *Database Integration Service* to replicate tables between two Oracle TimesTen DataStores.

Database Integration Service uses lazy replication to send updates to other hosts in the network. Using lazy replication, a database update is sent to the subscribers after the transaction is committed.

<NDDSHOME> is described in Paths Mentioned in Documentation (Section 1.1).

1. Follow the instructions in Section 2.1 through Section 2.5 on a second host.
2. Set the tag **<enable_table_replication>** in **<general_options>** to true in the configuration files (**RTI_REAL_TIME_CONNECT.xml** in this example) on host1 and host2.
3. Start *Database Integration Service* by running the following command from **<NDDSHOME>/bin**:

   ```
   host1> ./rtirtc_timesten -noDaemon -cfgName default
   host2> ./rtirtc_timesten -noDaemon -cfgName default
   ```

4. Start the TimesTen Interactive SQL utility from the command prompt:

   ```
   host1> ttlsql -connStr "DSN=Example"
   host2> ttlsql -connStr "DSN=Example"
   ```

5. Create a table named **ExampleRep** in host1 that will be replicated in host2:

   ```
   Command> CREATE TABLE ExampleRep (
            pkey TT_SMALLINT NOT NULL,
            message TT_VARCHAR(13),
            RTIDDS_DOMAIN_ID TT_INTEGER,
            RTIRTC_REMOTE TT_INTEGER,
            PRIMARY KEY (pkey));
   ```

   Note: RTIDDS_DOMAIN_ID and RTIRTC_REMOTE are meta-columns used by *Database Integration Service* to control the replication process.

6. To replicate the **ExampleRep** table in host 1 and 2, set up a publication and a subscription in each one of the hosts:

   On host1:

   ```
   Command> insert into RTIDDS_SUBSCRIPTIONS values
   ('Student', 'ExampleRep', 0, 'Example MyTypeRep', 'MyTypeRep');
   Command> insert into RTIDDS_PUBLICATIONS values
   ('Student', 'ExampleRep', 0, 'Example MyTypeRep', 'MyTypeRep');
   ```

   On host2:

   ```
   Command> insert into RTIDDS_SUBSCRIPTIONS values
   ('Student', 'ExampleRep', 0, 'Example MyTypeRep', 'MyTypeRep');
   Command> insert into RTIDDS_PUBLICATIONS values
   ('Student', 'ExampleRep', 0, 'Example MyTypeRep', 'MyTypeRep');
   ```

7. Verify that the table **ExampleRep** has been created automatically in host 2:

   ```
   Command> tables;
     STUDENT.EXAMPLEREP
     STUDENT.RTIDDS_PUBLICATIONS
     STUDENT.RTIDDS_SUBSCRIPTIONS
     STUDENT.RTIRTC_LOG
     STUDENT.RTIRTC_TBL_INFO
   ```

8. Insert a new row in host 1:

```
Command> insert into ExampleRep values (0, 'Hello World 0');
1 row inserted.
```

9. Verify that the row has been replicated in host 2:

```
Command> select * from ExampleRep;
< 0, Hello World 0, 0, 1 >
1 row found.
```

10. Modify the row in host 2:

```
Command> update ExampleRep set message = 'Hello World 1';
1 row updated.
```

11. Verify that the change has been propagated to host 1:

```
Command> select * from ExampleRep;
< 0, Hello World 1, 0, 123 >
 1 row found.
```

12. Delete the row in host 1:

```
Command> delete from ExampleRep;
1 row deleted.
```

13. Verify that the change has been propagated to host 2:

```
Command> select * from ExampleRep;
0 rows found.
```

Chapter 3 Database Integration Service for Oracle TimesTen on Windows Systems

This chapter provides instructions on how to install and use *Database Integration Service* for Oracle TimesTen on Windows platforms.

This chapter assumes you are using Microsoft Visual Studio® 2013[1].

3.1 Installation

Install *Database Integration Service* on top of *RTI Connext DDS*. There are two ways to install it, from *RTI Launcher* or the command line.

<NDDSHOME> is described in Paths Mentioned in Documentation (Section 1.1).

From RTI Launcher:

1. Start *RTI Launcher* from the Start menu or the command line:

```
cd <NDDSHOME>\bin
rtilauncher
```

2. From the Utilities tab, click on **RTI Package Installer**.
3. Use the + sign to add the **.rtipkg** file that you want to install.
4. Click **Install**.

From the command line:

```
cd <NDDSHOME>\bin
rtipkginstall <path to .rtipkg file>
```

1. You may use other supported Microsoft compilers listed in the *RTI Connext DDS Core Libraries Platform Notes*.

3.2 Creating an Oracle TimesTen Database

Before you can use *Database Integration Service* for Oracle TimesTen, you need to get a TimesTen user account to an existing data store (database) from your database administrator. This section explains how to create a TimesTen database when you act as your own database administrator.

1. Log in as the instance administrator (the user that installed TimesTen).
2. Create a DSN for the new TimesTen database as described in Creating a Data Source (Section 3.2.1).
3. Create the new datastore by executing the following[1]:

```
> ttlsql -connStr "DSN=ExampleAdmin"
Command> exit
```

3.2.1 Creating a Data Source

To add a data source:

1. Open the ODBC Data Source Administrator:
   - On Windows systems: Select **Start, Control Panel, System and Security, Administrative Tools, Data Sources (ODBC).**
2. Select the **User DSN** tab.
3. Click **Add...**; the **Create New Data Source** dialog appears.
4. Select **TimesTen Data Manager 11.2.1** from the list of drivers.
5. Click **Finish**; the **TimesTen ODBC Setup** dialog appears.
6. Fill in the fields in the dialog.
   a. Enter "**ExampleAdmin**" as the **Data Source Name** (DSN).
   b. Enter a **Data Store Path**; the path must exist, so you may want to create a new directory first.

      The data-store path name uniquely identifies the physical data-store. It is the full *path* name of the data-store (e.g., **C:\data\AdminData**). Note that this is not a *file* name—the actual data-store file names have suffixes (e.g., **C:\data\AdminData.ds0**, **C:\data\AdminData.log0**).
   c. Be sure the **Temporary** check box is *not* selected.
   d. Set **Database Character Set** to AL32UTF8.

      If you want to use *Database Integration Service* for Oracle TimesTen with Oracle In-Memory Database Cache (IMDB) (see Chapter 6: Database Integration Service for TimesTen with Oracle In-Memory Database Cache), make sure that the attribute **DatabaseCharacterSet** is set to the same character set used by Oracle 11g.

      You can determine the Oracle database character set by executing the following query on SQL*Plus (as any user, all on one line):

```
SQL> SELECT value FROM nls_database_parameters WHERE
 parameter='NLS_CHARACTERSET';
```

   e. All other fields can be left empty.

1. If *ttlsql* is not on your path, it can be found in the **bin** directory of your Oracle TimesTen installation.

7. Select the **IMDB Cache** tab if you want to integrate with Oracle In-Memory Database Cache.
   a. Enter the Oracle Net service Name. For example, “orcl11g”.
   b. Click **OK**.

3.3 Creating an Oracle TimesTen User Account

Starting with Oracle Timesten 11.2.1, database access control is mandatory.

If you are acting as your own database and system administrator, use the following steps to create an account for the user “Student”:

1. Log in as the instance administrator (the user that installed TimesTen).
2. Connect to the Oracle TimesTen database where the user account must be created:

```
> ttlsql -connStr "DSN=ExampleAdmin"
```

3. Enter the following to create a user account with the name “Student” and password “mypsswrd”:

```
Command> create user Student identified by mypsswrd;
Command> grant all privileges to Student;
Command> exit
```

3.4 Creating a Data Source for Oracle TimesTen

Database Integration Service for Oracle TimesTen uses the Oracle TimesTen ODBC driver to access data sources (datastores). In this section you will create a data source (DSN) for *Database Integration Service* for Oracle.

To add a data source:

1. Open the ODBC Data Source Administrator:
   - Select **Start, Control Panel, System and Security, Administrative Tools, Data Sources (ODBC).**
2. Select the **User DSN** tab.

   In this document *Database Integration Service* will be run from the command line. If you want to run the daemon as a Window service select **System DSN** instead of User DSN.
3. Click **Add**; the **Create New Data Source** dialog appears.
4. Select **TimesTen Data Manager 11.2.1** from the list of drivers.
5. Click **Finish**; the **TimesTen ODBC Setup** dialog appears.
6. Fill out the fields in the dialog.
   a. Enter “**Example**” as the **Data Source Name** (DSN).
   b. Enter a **Data Store Path**; the path must exist, so you may want to create a new directory first. Make sure that the value of this field is the same value that was used to create the DSN in Creating a Data Source (Section 3.2.1).

The data-store path name uniquely identifies the physical data-store. It is the full *path* name of the data-store (e.g., **C:\data\AdminData**). Note that this is not a **file** name—the actual data-store file names have suffixes (e.g., **C:\data\AdminData.ds0**, **C:\data\AdminData.log0**).

c. Be sure the **Temporary** check box is *not* selected.

d. Set **Database Character Set** to AL32UTF8.

If you want to use *Database Integration Service* for Oracle TimesTen with Oracle In-Memory Database Cache (IMDB) (see Chapter 6: Database Integration Service for TimesTen with Oracle In-Memory Database Cache), be sure that the attribute **DatabaseCharacterSet** is set to the same character set that Oracle 11g uses.

You can determine the Oracle database character set by executing the following query on SQL*Plus as any user (enter it all on one line):

```
SQL> SELECT value FROM nls_database_parameters WHERE
        parameter='NLS_CHARACTERSET';
```

e. All other fields can be left empty.

7. Select the **General Connection** tab and enter "**Student**" as the **User ID**.
8. Select the **IMDB Cache** tab if you want to integrate with Oracle In-Memory Database Cache.
   a. Enter the Oracle Net service Name. For example "orcl11g".
   b. Enter the Oracle Password of the user 'Student' in the Oracle 11g database.
9. Click **OK**.

3.5 Creating a Configuration File for Database Integration Service

Database Integration Service reads its configuration information from a file. By default, *Database Integration Service* tries to load the configuration file, **<NDDSHOME>\resource\xml\RTI_REAL_TIME_CONNECT.xml**. You can specify a different file with the command line option, **-cfgFile**.

The default file, **RTI_REAL_TIME_CONNECT.xml**, does not contain any valid database configuration information yet. For this example, we will edit this file as follows:

1. Look for the tag **<timesten_connection>**. Replace the tags **<dsn>. <user_name>,** and **<password>** as follows:

```
<timesten_connection>
   <dsn>Example</dsn>
   <user_name>Student</user_name>
   <password>mypsswrd</password>
</timesten_connection>
```

2. Save the file.

This configuration file instructs *Database Integration Service* to monitor the data source as specified by the "**Example**" DSN.

3.6 Using Database Integration Service as a Windows Service

3.6.1 Enabling Database Integration Service to Run as a Windows Service

If you want to run *Database Integration Service* for Oracle TimesTen as a Windows Service, you must install it as such before running it. To install it as a Windows Service, run the following command in a terminal with Administrator privileges:

```
<NDDSHOME>\bin\rtirtc_timesten -installService
```

3.6.2 Running Database Integration Service as a Windows Service

Windows Services automatically run in the background when the system reboots.

To run *Database Integration Service* for Oracle TimesTen as a Windows Service from the command line, use the Windows **sc** utility:

```
sc rtirtc_timesten530 start
```

Or you can run the application directly without the **-noDaemon** argument:

```
<NDDSHOME>\bin\rtirtc_timesten
```

To stop *Database Integration Service* for Oracle TimesTen when it is running as a Windows Service, use this command:

```
sc rtirtc_timesten530 stop
```

Alternatively, you can start/stop *Database Integration Service* for Oracle TimesTen from the Windows Services Control Manager. From the **Start** menu, select **Control Panel**, **Administrative Services**, **Services**. Click on the service in the list, then right-click to select **Start** or **Stop**.

3.6.3 Disabling Database Integration Service from Running as a Windows Service

To remove *Database Integration Service* for Oracle TimesTen from the list of Windows Services on your system, run this command in a terminal with Administrator privileges:

```
<NDDSHOME>\bin\rtirtc_timesten -uninstallService
```

3.7 Starting Database Integration Service for Oracle Daemon

You can start the *Database Integration Service* for Oracle Daemon as a Windows service (assuming you followed the steps in Using Database Integration Service as a Windows Service (Section 3.6)). However, for the following example, we will start the daemon manually.

Start *Database Integration Service* for Oracle TimesTen by executing the following command from the **<NDDSHOME>\bin** directory.

```
rtirtc_timesten -noDaemon -cfgName default
```

By default, *Database Integration Service* for Oracle TimesTen tries to start in the background as a Windows Service. (In order to do that, it must have been enabled as a Windows Service following the steps in Enabling Database Integration Service to Run as a Windows Service (Section 3.6.1).) However, if you specify the **-noDaemon** option, *Database Integration Service* Daemon starts as a regular process (instead of as a Windows Service). Messages are sent to standard output.

You should see output similar to the following, indicating that the process is running.

```
> rtirtc_timesten -noDaemon -cfgName default
RTI Database Integration Service for TimesTen, Release 5.x.y: startup succeeded
```

Database Integration Service for Oracle TimesTen is now connected to the "**Example**" data source.

3.8 Storing Samples from Publishers

In this section, you will enable automatic capturing of data samples in a Oracle TimesTen data store. The first step is to create an IDL type definition. By using *RTI Code Generator's*[1] **-example** option, you will automatically create a Publisher of this IDL type.

1. Create a new text file called **MyType.idl** with the following contents:

   ```
   struct MyType {
      short pkey; //@key
      char message[13];
   };
   ```

   This IDL file specifies a data type that contains a message. Each instance is uniquely identified by the **pkey** field.

2. Now execute the following command to compile the IDL type:

   ```
   rtiddsgen -language C -example i86Win32VS2013[2] -ppDisable MyType.idl
   ```

   This generates the **MyType**, **MyTypePlugin**, and **MyTypeSupport** files, as well as the **MyType_publisher** and **MyType_subscriber** example code.

3. The generated example will also have a Visual Studio solution file, **MyType-vs2013.sln**. Start Microsoft Visual Studio 2013 and load this solution by clicking on this file.

4. Edit **MyType_publisher.c**, and find the line containing the comment:

   ```
   /* Modify the data to be written here */
   ```

   Insert the following lines immediately below this comment:

   ```
   instance->pkey = count;
   strcpy(instance->message, "Hello world!");
   ```

5. Save your changes and build the **MyType_publisher** project in Visual Studio.

6. Start the **MyType_publisher** application so that it starts publishing data samples. From a command prompt,

   ```
   > objs\i86Win32VS2013\MyType_publisher
   ```

   On the screen, you will see:

   ```
   Writing MyType, count 0
   Writing MyType, count 1
   Writing MyType, count 2
   ...
   ```

   The samples are not captured in the Oracle TimesTen data store yet. For this, you need to set up a subscription in *Database Integration Service* for Oracle TimesTen.

1. *RTI Code Generator (rtiddsgen)* is an IDL code generator distributed with *Connext DDS*. Please see the *RTI Code Generator User's Manual* for more information on how to run *rtiddsgen*.
2. If you are using a different supported compiler, you will need to use a different value here, such as i86Win32VS2010 for Visual Studio 2010.

Subscriptions are set up in the **RTIDDS_SUBSCRIPTIONS** configuration table that *Database Integration Service* for Oracle TimesTen created when it connected to the Oracle TimesTen data store. For inserting entries into this table, you can use the Oracle TimesTen interactive SQL utility.

7. Start the TimesTen Interactive SQL utility from the command prompt:

```
> ttlsql -connStr "DSN=Example"
```

8. You can see that the **RTIDDS_SUBSCRIPTIONS** configuration table is still empty at this point by executing the following SQL command—*don't forget to type a semicolon '**;**' at the end of the line*:

```
Command> select * from RTIDDS_SUBSCRIPTIONS;
0 rows found.
```

9. To store the samples from the **MyType_publisher** application in a table named **Example**, you insert a corresponding entry into the **RTIDDS_SUBSCRIPTIONS** table:

```
Command> insert into RTIDDS_SUBSCRIPTIONS values
('Student', 'Example', 0, 'Example MyType', 'MyType');
1 row inserted.
```

This entry directs *Database Integration Service* for Oracle TimesTen to create a user table named **Example** with owner **Student** and to start storing samples published with topic **Example MyType** and data type **MyType** in Domain **0**.

10. If the **MyType_publisher** application is still running, you can execute the following SQL statement to view the contents of the table—otherwise, restart **MyType_publisher** as described above before executing this statement.

```
Command> select * from Example;
```

The output will look something like this:

```
...
< 134, Hello world!, 0, 1 >
< 135, Hello world!, 0, 1 >
< 136, Hello world!, 0, 1 >
85 rows found.
```

The actual number of rows found depends on when exactly the *Database Integration Service* Daemon started storing samples. If you execute the **select** statement repeatedly you will see the number of rows grow. This is because the **MyType_publisher** application writes a new instance every 4 seconds.

3.9 Publishing Database Updates to Subscribers

In this section, you will enable *Database Integration Service* for Oracle TimesTen to automatically publish changes made to a table in a data source. For this, you will again use the Oracle TimesTen Interactive SQL utility to change a record in the **Example** table. We assume that you have followed the instructions in Using Database Integration Service as a Windows Service (Section 3.6) to start a *Database Integration*

Service Daemon. In addition, we assume that you have followed the instructions in Steps 1 through 4 in Storing Samples from Publishers (Section 3.8) to create the IDL file and generate and compile the example code.

1. If you have not already done so, stop the **MyType_publisher** application by pressing **Ctrl-c** in the window where it is running.
2. Build the **MyType_subscriber** project in Visual Studio and start the application from the command line:

```
> objs\i86Win32VS2013\MyType_subscriber
MyType subscriber sleeping for 4 sec...
MyType subscriber sleeping for 4 sec...
...
```

   You will get this message repeatedly, since nothing is being published.

3. For publishing changes to a data source, you need to configure the **RTIDDS_PUBLICATIONS** table. For publishing the changes to the **Example** table, execute the following SQL statement:

```
Command> insert into RTIDDS_PUBLICATIONS values
('Student', 'Example', 0, 'Example MyType', 'MyType');
```

   This entry directs *Database Integration Service* for Oracle TimesTen to start publishing changes to table **Example** of owner **Student** as the topic **Example MyType** with type **MyType** in Domain **0**.

4. Now change one of the previously captured samples in the **Example** table, for instance the last one:

```
Command> update Example set message = 'Hello again!' where pkey in (select
max(pkey) from Example);
```

   You will see that the **MyType_subscriber** application reports the update, for example:

```
...
MyType subscriber sleeping for 4 sec...
pkey: 748
   message:
      message[ 0]: 'H'
      message[ 1]: 'e'
      message[ 2]: 'l'
      message[ 3]: 'l'
      message[ 4]: 'o'
      message[ 5]: ' '
      message[ 6]: 'a'
      message[ 7]: 'g'
      message[ 8]: 'a'
      message[ 9]: 'i'
      message[10]: 'n'
      message[11]: '!'
      message[12]: <0>
MyType subscriber sleeping for 4 sec...
...
```

5. You can also update all entries in the table:

```
Command> update Example set message = 'Hello again!';
```

   Notice that the **MyType_subscriber** application receives all changes, possibly a large number.

3.10 Database Table Replication

This section explains how to use *Database Integration Service* to replicate tables between two Oracle TimesTen DataStores.

Database Integration Service uses lazy replication to send updates to other hosts in the network. Using lazy replication, a database update is sent to the subscribers after the transaction is committed.

1. Follow the instructions in Creating an Oracle TimesTen Database (Section 3.2) through Creating a Configuration File for Database Integration Service (Section 3.5) on a second host.
2. Set the tag **<enable_table_replication>** in **<general_options>** to true in the configuration files on host1 and host2.
3. Start *Database Integration Service* by running the following command from **bin\i86Win32VS2013** in the installation directory:

```
host1> rtirtc_timesten -noDaemon -cfgName default
host2> rtirtc_timesten -noDaemon -cfgName default
```

4. Start the TimesTen Interactive SQL utility from the command prompt:

```
host1> ttlsql -connStr "DSN=Example"
host2> ttlsql -connStr "DSN=Example"
```

5. Create a table named **ExampleRep** in host 1 that will be replicated in host2:

```
Command> CREATE TABLE ExampleRep (
pkey TT_SMALLINT NOT NULL,
message TT_VARCHAR(13),
RTIDDS_DOMAIN_ID TT_INTEGER,
RTIRTC_REMOTE TT_INTEGER,
PRIMARY KEY (pkey));
```

   Note: RTIDDS_DOMAIN_ID and RTIRTC_REMOTE are meta-columns used by *Database Integration Service* to control the replication process.

6. To replicate the **ExampleRep** table in host 1 and 2, set up a publication and a subscription in each one of the hosts:

   On host1:

```
Command> insert into RTIDDS_SUBSCRIPTIONS values
('Student', 'ExampleRep', 0, 'Example MyTypeRep', 'MyTypeRep');
Command> insert into RTIDDS_PUBLICATIONS values
('Student', 'ExampleRep', 0, 'Example MyTypeRep', 'MyTypeRep');
```

   On host2:

```
Command> insert into RTIDDS_SUBSCRIPTIONS values
('Student', 'ExampleRep', 0, 'Example MyTypeRep', 'MyTypeRep');
Command> insert into RTIDDS_PUBLICATIONS values
('Student', 'ExampleRep', 0, 'Example MyTypeRep', 'MyTypeRep');
```

7. Verify that the table **ExampleRep** has been created automatically in host 2:

```
Command> tables;
  STUDENT.EXAMPLEREP
  STUDENT.RTIDDS_PUBLICATIONS
  STUDENT.RTIDDS_SUBSCRIPTIONS
  STUDENT.RTIRTC_LOG
  STUDENT.RTIRTC_TBL_INFO
```

8. Insert a new row on host 1:

```
Command> insert into ExampleRep values (0, 'Hello World 0');
1 row inserted.
```

9. Verify that the row has been replicated on host 2:

```
Command> select * from ExampleRep;
< 0, Hello World 0, 0, 1 >
1 row found.
```

10. Modify the row in host 2:

```
Command> update ExampleRep set message = 'Hello World 1';
1 row updated.
```

11. Verify that the change has been propagated to host 1:

```
Command> select * from ExampleRep;
< 0, Hello World 1, 0, 123 >
1 row found.
```

12. Delete the row on host 1:

```
Command> delete from ExampleRep;
1 row deleted.
```

13. Verify that the change has been propagated to host 2

```
Command> select * from ExampleRep;
0 rows found.
```

Chapter 4 Database Integration Service for Oracle 11g on UNIX-based Systems

4.1 Installation

First, verify that Oracle 11g Release 2 is installed and running on your system.

The installation of Oracle 11g is beyond the scope of this document. Please refer to Oracle documentation for the process to install and configure Oracle 11g.

1. Install *Database Integration Service* on top of *RTI Connext DDS*. There are two ways to install it, from *RTI Launcher* or from the command line.

   <NDDSHOME> is described in Paths Mentioned in Documentation (Section 1.1).

   To install from RTI Launcher:

   a. Start *RTI Launcher*:

   ```
   cd <NDDSHOME>/bin
   ./rtilauncher
   ```

   b. From the Utilities tab, click on **RTI Package Installer**.

   c. Use the + sign to add the **.rtipkg** file that you want to install.

   d. Click **Install**.

   To install from the command line:

   ```
   cd <NDDSHOME>/bin
   ./rtipkginstall <path to .rtipkg file>
   ```

2. Add the path to the Oracle 11g libraries to the beginning of **LD_LIBRARY_PATH**. For example:

   ```
   > setenv LD_LIBRARY_PATH /opt/oracle/product/11.2.0/dbhome_1/lib:$LD_LIBRARY_PATH
   ```

   or

   ```
   > export LD_LIBRARY_PATH=/opt/oracle/product/11.2.0/dbhome_1/lib:$LD_LIBRARY_PATH
   ```

3. Set ORACLE_HOME. For example:

   ```
   > setenv ORACLE_HOME /opt/oracle/product/11.2.0/dbhome_1
   ```

   or

   ```
   > export ORACLE_HOME=/opt/oracle/product/11.2.0/dbhome_1
   ```

4.2 Configuring the Oracle Server

Database Integration Service Daemon uses external procedures, EXTPROCs, to interface with the Oracle server. Or more accurately, *Database Integration Service* provides external procedures in a library, **librtirtc_oracleq.so**, that the Oracle server must be able to load while running to communicate with the *Database Integration Service* Daemon. Chapter 4 in the *Database Integration Service User's Manual* provides a detailed process to follow in order to install and configure the Oracle server to use **librtirtc_oracleq.so**.

This section provides a procedure with the least amount of configuration. Alternative ways to install the files required to support *Database Integration Service*'s external procedures can be found in the accompanying *Database Integration Service User's Manual.*

4.2.1 Install librtirtc_oracleq.so

Copy the appropriate version of **librtirtc_oracleq.so** into **$ORACLE_HOME/bin** on the server host. **$ORACLE_HOME** is the installation directory of the Oracle 11g database. This library is distributed with *Database Integration Service* for Oracle and can be found in **lib/*<arch>*** in the installation directory. The correct version of the library to use depends on the platform on which the Oracle server is running. For example, <*arch*> might be i86Linux2.6gcc4.1.1.

You can also configure the Oracle server to find the shared library in directories outside of the Oracle installation tree. Please see Chapter 4 in the *Database Integration Service User's Manual*.

4.2.2 Install libnddsc.so and libnddscore.so

This section only applies to Linux systems.

librtirtc_oracleq.so uses *Connext DDS* and thus the shared libraries **libnddsc.so** and **libnddscore.so** must also be installed and be accessible at runtime by the Oracle server.

Add the directory containing the appropriate *Connext DDS* libraries to the environment variable, **LD_LIBRARY_PATH**, for the user who starts the Oracle server. You may need to restart the Oracle server after this variable has been changed.

Other ways of installing **libnddsc.so** and **libnddscore.so** on the Oracle server can be found in Chapter 4 in the *Database Integration Service User's Manual*.

Note: If you are using a license managed version of *RTI Connext DDS* (e.g., an evaluation installer), make sure you define the RTI_LICENSE_FILE environment variable to point to a valid license file. Also make sure that this environment variable is visible from the Oracle Database server process. In order to take effect, on some systems you may need to define it as a system-wide environment variable and restart the Oracle database process. You can find more information about using a license file in Section 2.4 of the *RTI Connext DDS Core Libraries Getting Started Guide*.

4.2.3 Create an Oracle Account

Before you can use *Database Integration Service* for Oracle, you need to obtain an Oracle account from your database administrator. If you are acting as your own database administrator, start **SQL*Plus** from the command prompt on the Oracle server and log in as the system manager:

```
> sqlplus system/<password>@<Oracle_Service_Name>
```

For example, to create a new Oracle account for "**Student**" on the Oracle database identified by the connection string "**orcl11g**" with system manager password "**mypsswrd**", enter the following:[1]

1. If *sqlplus* is not on your path, it can be found in the **bin/** directory of your Oracle installation.

```
> sqlplus system/mypsswrd@orcl

SQL> CREATE USER Student IDENTIFIED BY mypsswrd DEFAULT TABLESPACE users;
SQL> GRANT connect, resource, create any trigger, create any library, create any
table TO Student;
SQL> COMMIT;
SQL> EXIT
```

The remaining sections in this chapter assume that an Oracle user named "**Student**" with the password "**mypsswrd**" has an account for the "**orcl11g**" database.

4.3 Creating a Data Source for Oracle

Database Integration Service for Oracle usually connects to the same data source or data sources to which your applications connect. The connection information for each data source is stored in the "**.odbc.ini**" file. The stored information describes each data source in detail, specifying the driver name, a description, and any additional information the driver needs to connect to the data source.

To create the "**.odbc.ini**" file, follow these steps:

1. Create a new file named "**.odbc.ini**" in your home directory using your favorite text editor. Alternatively, you can use the **ODBCINI** environment variable to specify the file location.
2. Insert these lines in the file:

```
[ODBC Data Source]
Example=Oracle Driver
[Example]
DRIVER=/opt/oracle/product/11.2.0/dbhome_1/lib/libsqora.so.11.1
Servername=ORCL11G
```

   NOTE Make sure that **DRIVER** points to the valid location of the Oracle driver on your system.

3. Save your changes.

4.4 Creating a Configuration File for Database Integration Service

Database Integration Service reads its configuration information from a file. By default, *Database Integration Service* tries to load the configuration file, **<NDDSHOME[1]>/resource/xml/RTI_REAL_TIME_CONNECT.xml**. You can specify a different file with the command line option, **-cfgFile**.

The default file, **RTI_REAL_TIME_CONNECT.xml**, does not contain any actual database configuration information yet. For this example we will edit this file as follows:

1. Look for the tag **<oracle_connection>**. Replace the tags **<dsn>, <user_name>,** and **<password>** as follows:

```
<oracle_connection>
  <dsn>Example</dsn>
  <user_name>Student</user_name>
  <password>mypsswrd</password>
</oracle_connection>
```

2. Save the file.

1. See Paths Mentioned in Documentation (Section 1.1)

This configuration file instructs *Database Integration Service* to monitor the data source as specified by the "Example" DSN.

4.5 Starting the Database Integration Service for Oracle Daemon

Start *Database Integration Service* for Oracle:

```
> cd <NDDSHOME[1]>/bin
> ./rtirtc_oracle -noDaemon -cfgName default
```

By default, *Database Integration Service* for Oracle runs in the background as a daemon process. Specifying the "**-noDaemon**" option prevents that, and starts up the *Database Integration Service* Daemon as a regular process. Messages are sent to standard output.

You should see the following output:

```
>./rtirtc_oracle -noDaemon -cfgName default
RTI Database Integration Service for Oracle, Release 5.x.y: startup succeeded
```

Database Integration Service for Oracle is now connected to the "**Example**" data source.

Note: Make sure that you have the Oracle 11g libraries on your **LD_LIBRARY_PATH**, see Installation (Section 4.1).

4.6 Storing Samples from Publishers

In this section, you will enable automatic capturing of data samples in an Oracle 11g database. The first step is to create an IDL type definition. By using the "**-example**" option of *rtiddsgen*[2] you will automatically create a Publisher of this IDL type.

1. Create a new text file called "**MyType.idl**" with the following contents:

   ```
   struct MyType {
       short pkey; //@key
       char message[13];
   };
   ```

   This IDL file specifies a data type that contains a message. Each instance is uniquely identified by the "**pkey**" field.

2. Now execute the following command to compile the IDL type:

   ```
   > rtiddsgen -language C -example <arch> MyType.idl
   ```

   For example:

   ```
   > rtiddsgen -language C -example i86Linux2.6gcc4.1.1 MyType.idl
   ```

   This generates the **MyType**, **MyTypePlugin**, and **MyTypeSupport** files, as well as the **MyType_publisher** and **MyType_subscriber** example code.

3. The generated example will also have a makefile named:

   ```
   makefile_MyType_<arch>
   ```

1. See Paths Mentioned in Documentation (Section 1.1)
2. *rtiddsgen* is an IDL code generator distributed with *Connext DDS*. Please refer to the *RTI Code Generator User's Manual* for more information about how to run *rtiddsgen*.

You may need to edit the makefile to specify the location of the compiler if it is not available on your path.

4. Edit **MyType_publisher.c**, and find the line containing the comment:

```
/* Modify the data to be written here */
```

Insert the following lines immediately below this comment:

```
instance->pkey = count;
strcpy(instance->message, "Hello world!");
```

5. Save your changes and build the **MyType_publisher** and **MyType_subscriber** applications by executing:

```
> gmake -f makefile_MyType_<arch>
```

6. Start the **MyType_publisher** application so that it starts publishing data samples.

```
> objs/<arch>/MyType_publisher
```

On the screen, you will see:

```
Writing MyType, count 0
Writing MyType, count 1
Writing MyType, count 2
...
```

The samples are not captured in the Oracle 11g database yet. For this you need to set up a subscription in *Database Integration Service* for Oracle.

Subscriptions are set up in the "**RTIDDS_SUBSCRIPTIONS**" configuration table that *Database Integration Service* for Oracle created when it connected to the Oracle database. To insert entries into this table, you can use the Oracle's **SQL*Plus** utility again.

7. Start the **SQL*Plus** utility from the command prompt:[1]

```
> sqlplus student/mypsswrd@orcl11g
SQL> set autocommit on;[2]
```

8. You can see that the "**RTIDDS_SUBSCRIPTIONS**" configuration table is still empty at this point by executing the following SQL command—*don't forget to type a semicolon ';' at the end of the line*:

```
SQL> select * from RTIDDS_SUBSCRIPTIONS;
0 rows found.
```

9. To store the samples from the **MyType_publisher** application in a table named "**Example**", you insert a corresponding entry into the "**RTIDDS_SUBSCRIP-TIONS**" table:

```
SQL> insert into RTIDDS_SUBSCRIPTIONS (table_owner, table_name, domain_id, top-
ic_name, type_name) values
('Student', 'Example', 0, 'Example MyType', 'MyType');
1 row inserted.
```

This entry directs *Database Integration Service* for Oracle to create a user table named "**Example**" with owner "**Student**" and to start storing samples published with topic "**Example MyType**" and data type "**MyType**" in Domain **0**.

1. If *sqlplus* is not on your path, it can be found in the **bin/** directory of your Oracle installation.
2. By default, *sqlplus* is not in autocommit mode. Use this statement to turn on autocommit.

10. If the **MyType_publisher** application is still running, you can execute the following SQL statement to view the contents of the table—otherwise, restart **MyType_publisher** as described above before executing this statement.

```
SQL> select * from Example;
```

The output will look something like this:

```
      PKEY MESSAGE       RTIDDS_DOMAIN_ID RTIRTC_REMOTE
---------- ------------- ---------------- -------------
        90 Hello world!                44             1
        91 Hello world!                44             1
        92 Hello world!                44             1
       ...
```

The actual number of rows found depends on exactly when the *Database Integration Service* Daemon started storing samples. If you execute the "**select**" statement repeatedly, you will see the number of rows grow. This is because the **MyType_publisher** application writes a new instance every 4 seconds.

4.7 Publishing Database Updates to Subscribers

In this section, you will enable *Database Integration Service* for Oracle to automatically publish changes made to a table in a data source. For this, you will again use the Oracle's **SQL*Plus** utility to change a record in the "**Example**" table.

We assume that you have followed the instructions in Starting the Database Integration Service for Oracle Daemon (Section 4.5) to start a *Database Integration Service* Daemon. In addition, we assume that you have followed the instructions in Steps 1 to 4 in Storing Samples from Publishers (Section 4.6) to create the IDL file and generate and compile the example code.

1. If you have not already done so, stop the **MyType_publisher** application by pressing "**Ctrl-c**" in the window where it is running.
2. Start the **MyType_subscriber** application from the command line:

```
> objs/<arch>/MyType_subscriber
MyType subscriber sleeping for 4 sec...
MyType subscriber sleeping for 4 sec...
...
```

You will get this message repeatedly, since nothing is being published.

3. For publishing changes to a data source, you need to configure the "**RTIDDS_PUBLICATIONS**" table. For publishing the changes to the "**Example**" table, execute the following SQL statement:

```
SQL> insert into RTIDDS_PUBLICATIONS (table_owner, table_name, domain_id, top-
ic_name, type_name) values
('Student', 'Example', 0, 'Example MyType', 'MyType');
```

This entry directs *Database Integration Service* for Oracle to start publishing changes to table "**Example**" of owner "**Student**" as the topic "**Example MyType**" with type "**MyType**" in the Domain **0**.

4. Now change one of the previously captured samples in the "**Example**" table, such as the last one:

```
SQL> update Example set message = 'Hello again!' where pkey in (select max(pkey)
from Example);
```

You will see that the **MyType_subscriber** application reports the update, for example:

```
...
MyType subscriber sleeping for 4 sec...
pkey: 748
   message:
      message[ 0]: 'H'
      message[ 1]: 'e'
      message[ 2]: 'l'
      message[ 3]: 'l'
      message[ 4]: 'o'
      message[ 5]: ' '
      message[ 6]: 'a'
      message[ 7]: 'g'
      message[ 8]: 'a'
      message[ 9]: 'i'
      message[10]: 'n'
      message[11]: '!'
      message[12]: <0>
MyType subscriber sleeping for 4 sec...
...
```

5. You can also update all entries in the table:

```
SQL> update Example set message = 'Hello again!';
```

Notice that the **MyType_subscriber** application receives all changes, possibly a large number.

Chapter 5 Database Integration Service for Oracle 11g on Windows Systems

This chapter provides instructions on installing and using *Database Integration Service* for Oracle 11g on Windows platforms.

This chapter assumes you will be *compiling* with Microsoft Visual Studio® 2013[1].

5.1 Before Installation

Before you can run *Database Integration Service*, you will also need:

- Oracle 11g Release 2 (must be installed and running).

  The installation of Oracle 11g is beyond the scope of this document. Please see Oracle documentation for how to install and configure Oracle 11g.

- Visual C++ Redistributable for Visual Studio 2010.

  It is available from this Microsoft website:
  http://www.microsoft.com/en-ca/download/details.aspx?id=30679.

5.2 Installation

Install *Database Integration Service* on top of *RTI Connext DDS*. There are two ways to install it, from *RTI Launcher* or from the command line.

<NDDSHOME> is described in Paths Mentioned in Documentation (Section 1.1).

From RTI Launcher:

1. Start *RTI Launcher* from the Start menu or from the command line:

   ```
   cd <NDDSHOME>\bin
   rtilauncher
   ```

2. From the Utilities tab, click on **RTI Package Installer**.
3. Use the + sign to add the **.rtipkg** file that you want to install.

1. You may use other supported Microsoft compilers such as Visual Studio 2010 or Visual Studio 2015, however, the instructions in this chapter were written assuming Visual Studio 2013.

4. Click **Install**.

From the command line:

```
cd <NDDSHOME>\bin
rtipkginstall <path to .rtipkg file>
```

5.3 Configuring the Oracle Server

The *Database Integration Service* Daemon uses external procedures, EXTPROCs, to interface with the Oracle server. Or more accurately, *Database Integration Service* provides external procedures in a library, **rtirtc_oracleq.dll**, that the Oracle server must be able to load while running to communicate with the *Database Integration Service* Daemon. Chapter 4 in the *Database Integration Service User's Manual* provides a detailed process to follow in order to install and configure the Oracle server to use **rtirtc_oracleq.dll**.

This section provides a procedure with the least amount of configuration. Alternative ways to install the files required to support *Database Integration Service*'s external procedures can be found in the *Database Integration Service User's Manual*.

5.3.1 Install rtirtc_oracleq.dll

Make sure that the environment variable **ORACLE_HOME** exists and is set correctly to the directory containing the installation of Oracle.

Copy the appropriate version of **rtirtc_oracleq.dll** into **%ORACLE_HOME%\bin** on the server host. **%ORACLE_HOME%** is the installation directory of the Oracle 11g database. This library is distributed with *Database Integration Service* for Oracle and can be found in **lib\<*architecture*>** in the installation directory.

You can also configure the Oracle server to find the shared library in directories outside of the Oracle installation tree. Please consult Chapter 4 in the *Database Integration Service User's Manual*.

5.3.2 Install nddsc.dll and nddscore.dll

Because **rtirtc_oracleq.dll** uses *Connext DDS*, the shared libraries **nddsc.dll** and **nddscore.dll** must also be installed and accessible at run time by the Oracle server. For that purpose, also make sure the permissions of the folder containing the **nddsc.dll** and **nddscore.dll** libraries are appropriate for the user who is running the Oracle server.

Important Notes:

- Make sure your **Path** system environment variable[1] contains the path to the *Connext DDS* libraries (**<NDDSHOME>\lib\<*arch*>,** where *<arch>* depends on your architecture, such as i86Win32VS2015). If **<NDDSHOME>\lib\<*arch*>** is not already in your **Path**, add it now. You will need to restart your computer after you modify **Path**. If you do not want to restart the computer, you can copy the libraries into a directory that is already in the system **Path**, such as **c:\Windows\System32**.
- If you are using a license managed version of *RTI Connext DDS* (e.g., an evaluation installer), make sure you define the RTI_LICENSE_FILE environment variable to point to a valid license file. Also make sure that this environment variable is visible from the Oracle Database server pro-

1. To view and/or edit the **Path** system environment variable: from the **Start** button/menu, select **Settings, Control Panel, System, Advanced tab,** then select the **Environment Variables** button. View or edit the **Path** in the System Variables. (The exact steps for accessing the environment variables may vary, depending on your version of the Windows operating system.)

cess. In order to take effect, on some systems you may need to define it as a system-wide environment variable and restart the Oracle database process. You can find more information about using a license file in Section 2.4 of the *RTI Connext DDS Core Libraries Getting Started Guide*.

5.3.3 Create an Oracle Account

Before you can use *Database Integration Service for Oracle*, you need to obtain an Oracle account from your database administrator. If you are acting as your own database administrator, start **SQL*Plus** from the command prompt on the Oracle server and log in as the system manager:[1]

```
> sqlplus system/<password>@<Oracle_Service_Name>
```

For example, to create a new Oracle account for "**Student**" on the Oracle database identified by the connection string "**orcl**" with system manager password "**mypsswrd**", enter the following:

```
> sqlplus system/mypsswrd@orcl
SQL> CREATE USER Student IDENTIFIED BY mypsswrd DEFAULT TABLESPACE users;
SQL> GRANT connect, resource, create any trigger, create any library, create any
table TO Student;
SQL> COMMIT;
SQL> EXIT
```

The remaining sections in this chapter assume that an Oracle user named "**Student**" with the password "**mypsswrd**" has an account for the "**orcl**" database.

5.4 Creating a Data Source for Oracle

Database Integration Service for Oracle 11g usually connects to the same data source or data sources to which your applications connect. The connection information for each data source is stored in the Windows registry. The stored information describes each data source in detail, specifying the driver name, a description, and any additional information the driver needs to connect to the data source.

To add a data source, follow these steps:

1. Open the ODBC Data Source Administrator:
   - On Windows systems: Select **Start, Control Panel, System and Security, Administrative Tools, Data Sources (ODBC).**
2. Select the **User DSN** tab.
3. Click the **Add** button; the **Create New Data Source** dialog appears.
4. Select the **Oracle 11g** driver from the list of drivers.
5. Click the **Finish** button; the **Oracle ODBC Driver Configuration** dialog appears.
6. Fill out the fields in the dialog.
   a. Enter "**Example**" as the **Data Source Name** (DSN).
   b. Select the **TNS Service Name**, and enter "**Student**" as the User ID.
   c. All other fields can be left empty.
7. Click the **OK** button.

1. If *sqlplus* is not on your path, it can be found in the **bin** directory of your Oracle installation.

5.5 Creating a Configuration File for Database Integration Service

Database Integration Service reads its configuration information from a file. By default, *Database Integration Service* tries to load the configuration file, **<NDDSHOME>/resource/xml/RTI_REAL_TIME_CONNECT.xml**. You can specify a different configuration file using the command line option, **-cfgFile**.

The default file **RTI_REAL_TIME_CONNECT.xml** does not contain any valid database configuration information yet. For this example we will edit this file as follows:

1. Look for the tag **<oracle_connection>**. Replace the tags **<dsn>, <user_name>,** and **<password>** as follows:

```
<oracle_connection>
         <dsn>Example</dsn>
         <user_name>Student</user_name>
         <password>mypsswrd</password>
 </oracle_connection>
```

2. Save the file.

This configuration file instructs *Database Integration Service* to monitor the data source as specified by the "Example" DSN.

5.6 Using Database Integration Service as a Windows Service

5.6.1 Enabling Database Integration Service to Run as a Windows Service

If you want to run *Database Integration Service* for Oracle 11g as a Windows Service, you must install it as such before running it. To install it as a Windows Service, run the following command in a terminal with Administrator privileges:

```
<NDDSHOME>\bin\rtirtc_oracle -installService
```

5.6.2 Running Database Integration Service as a Windows Service

Windows Services automatically run in the background when the system reboots.

To run *Database Integration Service* for Oracle 11g as a Windows Service from the command line, use the Windows **sc** utility:

```
sc rtirtc_oracle530 start
```

Or you can run the application directly without the **-noDaemon** argument:

```
<NDDSHOME>\bin\rtirtc_oracle
```

To stop *Database Integration Service* for Oracle 11g when it is running as a Windows Service, use this command:

```
sc rtirtc_oracle530 stop
```

Alternatively, you can start/stop *Database Integration Service* for Oracle 11g from the Windows Services Control Manager. From the **Start** menu, select **Control Panel**, **Administrative Services**, **Services**. Click on the service in the list, then right-click to select **Start** or **Stop**.

5.6.3 Disabling Database Integration Service from Running as a Windows Service

To remove *Database Integration Service* for Oracle 11g from the list of Windows Services on your system, run this command in a terminal with Administrator privileges:

```
<NDDSHOME>\bin\rtirtc_oracle -uninstallService
```

5.7 Starting the Database Integration Service for Oracle Daemon

You can start the *Database Integration Service* for Oracle Daemon as a Windows service (assuming you followed the steps in Using Database Integration Service as a Windows Service (Section 5.6)). However, for the following example, we will start the daemon manually.

Start *Database Integration Service* for Oracle by executing the following command from the **<NDDSHOME>\bin** directory.

```
rtirtc_oracle -noDaemon -cfgName default -dbtransport 1
```

By default, *Database Integration Service* for Oracle runs in the background as Windows service. Specifying the **-noDaemon** option prevents that, and the *Database Integration Service* Daemon is started as a regular process. Messages are sent to standard output.

You should see output similar to the following, indicating that the process is running.

```
> rtirtc_oracle -noDaemon -cfgName default
RTI Database Integration Service, Release 5.x.y: startup succeeded
```

Database Integration Service for Oracle is now connected to the "**Example**" data source.

5.8 Storing Samples from Publishers

In this section, you will enable automatic capturing of data samples in an Oracle 11g database. The first step is to create an IDL type definition. By using the "**-example**" option of *rtiddsgen*[1] you will automatically create a Publisher of this IDL type.

1. Create a new file called "**MyType.idl**" with the following contents:

```
struct MyType {
    short pkey; //@key
    char message[13];
};
```

   This IDL file specifies a data type that contains a message. Each instance is uniquely identified by the "**pkey**" field.

2. Now execute the following command to compile the IDL type:

```
> rtiddsgen -language C -example i86Win32VS2013[2] -ppDisable MyType.idl
```

   This generates the **MyType**, **MyTypePlugin**, and **MyTypeSupport** files, as well as the **MyType_publisher** and **MyType_subscriber** example code.

1. *rtiddsgen* is an IDL code generator distributed with *Connext DDS*. Please refer to the *RTI Code Generator User's Manual* for more information about how to run *rtiddsgen*.
2. If you are using a different supported compiler, you will need to use a different architecture name here. See the *RTI Connext DDS Core Libraries Release Notes*.

3. The generated example will also have a Visual Studio Solution file called **MyType-vs2013.sln**. Start Microsoft Visual Studio 2013 and load this solution by clicking on this file.

4. Edit **MyType_publisher.c**, and find the line containing the comment:

```
/* Modify the data to be written here */
```

   Insert the following lines immediately below this comment:

```
instance->pkey = count;
strcpy(instance->message, "Hello world!");
```

5. Save your changes and build the **MyType_publisher** project in Visual Studio.

6. Start the **MyType_publisher** application so that it starts publishing data samples. From a command prompt,

```
> objs\i86Win32VS2013\MyType_publisher
```

   On the screen, you will see:

```
Writing MyType, count 0
Writing MyType, count 1
Writing MyType, count 2
...
```

   The samples are not captured in the Oracle database yet. You still need to set up a subscription in *Database Integration Service* for Oracle.

   Subscriptions are set up in the "**RTIDDS_SUBSCRIPTIONS**" configuration table that *Database Integration Service* for Oracle created when it connected to the Oracle database. For inserting entries into this table you can use the Oracle's **SQL*Plus** utility again.

7. Start the **SQL*Plus** utility from the command prompt:[1]

```
> sqlplus student/mypsswrd@orcl
SQL> set autocommit on;[2]
```

8. You can see that the "**RTIDDS_SUBSCRIPTIONS**" configuration table is still empty at this point by executing the following SQL command—*don't forget to type a semicolon '**;**' at the end of the line*:

```
SQL> select * from RTIDDS_SUBSCRIPTIONS;
no rows selected.
```

9. To store the samples from the **MyType_publisher** application in a table named "**Example**", you insert a corresponding entry into the "**RTIDDS_SUBSCRIP-TIONS**" table:

```
SQL> insert into RTIDDS_SUBSCRIPTIONS (table_owner, table_name, domain_id, top-
ic_name, type_name) values
('Student', 'Example', 0, 'Example MyType', 'MyType');
1 row inserted.
```

   This entry directs *Database Integration Service* for Oracle to create a user table named "**Example**" with owner "**Student**" and to start storing samples published with topic "**Example MyType**" and data type "**MyType**" in Domain **0**.

10. If the **MyType_publisher** application is still running, you can execute the following SQL statement to view the contents of the table—otherwise, restart **MyType_publisher** as described above before executing this statement.

```
SQL> select * from Example;
```

1. If *sqlplus* is not on your path, it can be found in the **bin** directory of your Oracle installation.
2. By default, *sqlplus* is not in autocommit mode. Use this statement to turn on autocommit.

The output will look something like this:

```
PKEY MESSAGE       RTIDDS_DOMAIN_ID RTIRTC_REMOTE
---------- ------------- ---------------- -------------
        90 Hello world!                44             1
        91 Hello world!                44             1
        92 Hello world!                44             1
...
```

The actual number of rows found depends on when exactly the *Database Integration Service* Daemon started storing samples. If you execute the "**select**" statement repeatedly you will see the number of rows grow. This is because the **MyType_publisher** application writes a new instance every 4 seconds.

5.9 Publishing Database Updates to Subscribers

In this section, you will enable *Database Integration Service* for Oracle to automatically publish changes made to a table in a data source. For this, you will again use the Oracle's **SQL*Plus** utility to change a record in the "**Example**" table. We assume that you have followed the instructions in Starting the Database Integration Service for Oracle Daemon (Section 5.7) to start a *Database Integration Service* Daemon. In addition, we assume that you have followed the instructions in Steps 1 to 4 in Storing Samples from Publishers (Section 5.8) in creating the IDL file and generating and compiling the example code.

1. If you have not already done so, stop the **MyType_publisher** application by pressing "**Ctrl-c**" in the window where it is running.
2. Build the **MyType_subscriber** project in Visual Studio and start the application from the command line:

   ```
   > objs\i86Win32VS2013\MyType_subscriber
   MyType subscriber sleeping for 4 sec...
   MyType subscriber sleeping for 4 sec...
   ...
   ```

   You will get this message repeatedly, since nothing is being published.
3. For publishing changes to a data source, you need to configure the "**RTIDDS_PUBLICATIONS**" table. For publishing the changes to the "**Example**" table, execute the following SQL statement:

   ```
   SQL> insert into RTIDDS_PUBLICATIONS (table_owner, table_name, domain_id, topic_name, type_name) values
   ('Student', 'Example', 0, 'Example MyType', 'MyType');
   ```

   This entry directs *Database Integration Service* for Oracle to start publishing changes to table "**Example**" of owner "**Student**" as the topic "**Example MyType**" with type "**MyType**" in Domain **0**.
4. Now change one of the previously captured samples in the "**Example**" table, for instance the last one:

   ```
   SQL> update Example set message = 'Hello again!' where pkey in (select max(pkey)
   from Example);
   ```

   You will see that the **MyType_subscriber** application reports the update, for example:

   ```
   ...
   MyType subscriber sleeping for 4 sec...
   pkey: 748
   ```

```
   message:
      message[ 0]: 'H'
      message[ 1]: 'e'
      message[ 2]: 'l'
      message[ 3]: 'l'
      message[ 4]: 'o'
      message[ 5]: ' '
      message[ 6]: 'a'
      message[ 7]: 'g'
      message[ 8]: 'a'
      message[ 9]: 'i'
      message[10]: 'n'
      message[11]: '!'
      message[12]: <0>
MyType subscriber sleeping for 4 sec...
...
```

5. You can also update all entries in the table:

```
SQL> update Example set message = 'Hello again!';
```

Notice that the **MyType_subscriber** application receives all changes, possibly a large number.

Chapter 6 Database Integration Service for TimesTen with Oracle In-Memory Database Cache

This chapter describes how to install and run *Database Integration Service* for Oracle TimesTen with Oracle In-Memory Database Cache (IMDB).

For detailed information about Oracle In-Memory Database Cache, see the *Oracle In-Memory Database Cache User's Guide.*

This chapter assumes that you are familiar with IMDB and that you have already installed *Database Integration Service* for Oracle TimesTen and run through the exercises in Chapter 2: Database Integration Service for Oracle TimesTen on UNIX-Based Systems or Chapter 3: Database Integration Service for Oracle TimesTen on Windows Systems. If you have not yet done this, please do so first.

6.1 Installation

Oracle In-Memory Database Cache requires the installation of Oracle 11g and Oracle TimesTen 11.2.1.

The installation of Oracle 11g is beyond the scope of this document. Please refer to Oracle documentation for the process to install and configure Oracle 11g.

For instructions on how to install Oracle TimesTen 11.2.1, refer to Installation (Section 2.1).

If you did not configure the environment variable TNS_ADMIN during the Oracle TimesTen installation process, you can do it now using the **ttmodinstall** utility with the **-tns_admin** option. See the Oracle TimesTen In-Memory Database reference for more details.

```
> ttmodinstall -tns_admin /opt/oracle/product/11.2.0/dbhome_1/network/admin
```

The option **-tns_admin** is used to specify the location of the **tnsnames.ora** file in your Oracle 11g installation.

6.2 Creating Oracle Accounts

Before you can use *Database Integration Service* for Oracle TimesTen with Oracle In-Memory Database Cache, you need to create three Oracle users:

- **A Student user that owns the Oracle tables that will be cached in TimesTen. If you have not created the user yet,** Configuring the Oracle Server (Section 4.2) **(for UNIX-based systems) and** Configuring the Oracle Server (Section 5.3) **(for Windows systems) explain how to do it.**

- A **timesten** user that owns the Oracle tables used to store information about IMDB.
- A cache administration user, **cacheuser**, that owns and maintains Oracle objects that store information used to managed IMDB.

To create the **timesten** and **cacheuser** users, enter the following. We assume that the database is identified by the service name **orcl11g** and the sys password is **mypsswrd**.

```
> cd <Oracle TimesTen install dir>/oraclescripts
> sqlplus[1] sys/mypsswrd@orcl11g as sysdba

SQL> CREATE TABLESPACE cachetblsp DATAFILE 'datfttuser.dbf' SIZE 100M;
SQL> @initCacheGlobalSchema "cachetblsp"
SQL> CREATE USER cacheuser IDENTIFIED BY oracle DEFAULT TABLESPACE cachetblsp
QUOTA UNLIMITED ON cachetblsp;
SQL> @grantCacheAdminPrivileges "cacheuser"
SQL> exit
```

6.3 Creating an Oracle TimesTen Database

Before you can use *Database Integration Service* for Oracle TimesTen with Oracle In-Memory Database Cache, you need to create or gain access to a data store (database) with the same character set as the Oracle database. Creating an Oracle TimesTen Database (Section 2.2) (for UNIX-based systems) and Creating an Oracle TimesTen Database (Section 3.2) (for Windows systems) explain how to create a data store with Oracle TimesTen when you act as your own database administrator.

6.4 Creating Oracle TimesTen Accounts

In addition to the creation of Oracle users, integration with IMDB requires the creation of two Oracle TimesTen users:

- **A Student user that owns the cached tables in Oracle TimesTen. If you have not created the user yet,** Creating an Oracle TimesTen User Account (Section 2.3) **(for UNIX systems) and** Creating a Data Source (Section 3.2.1) **(for Windows systems) explain how to do it.**
- A cache manager user, **cacheuser**, that performs the cache group operations.

To create and configure the **cacheuser** user with password **timesten**, log in as the Oracle TimesTen administrator (the user that installed TimesTen) and follow these steps:

1. Connect to the Oracle TimesTen database created in Creating an Oracle TimesTen Database (Section 6.3):

   ```
   > ttlsql -connStr "DSN=Example"
   ```

2. Enter the following to create the user account:

   ```
   Command> create user cacheuser identified by timesten;
   Command> GRANT CREATE SESSION, CACHE_MANAGER, CREATE ANY TABLE TO cacheuser;
   Command> call ttCacheUidPwdSet('cacheuser','oracle');
   Command> exit
   ```

   The call to the **ttCacheUidPwdSet** built-in procedure sets the Oracle cache administration user name and password.

1. If *sqlplus* is not on your path, it can be found in the **bin/** directory of ORACLE_HOME.

3. As the new cache manager user, create a cache grid:

```
> ttlsql "DSN=Example;UID=cacheuser;PWD=timesten;OraclePWD=oracle"
Command> call ttGridCreate('myGrid');
Command> call ttGridNameSet('myGrid');
Command> exit
```

6.5 Creating a Data Source for Oracle In-Memory Database Cache

Database Integration Service for Oracle TimesTen with Oracle In-Memory Database Cache will use the TimesTen ODBC driver to access data sources (data stores). If you have not done it yet, Creating a Data Source for Oracle TimesTen (Section 2.4) (for UNIX-based systems) and Creating a Data Source for Oracle TimesTen (Section 3.4) (for Windows systems) explain how to create a data source for Oracle TimesTen that also works with IMDB.

6.6 Starting Database Integration Service for Oracle TimesTen

Before you start *Database Integration Service* for Oracle TimesTen, verify that your configuration file, **<RTIRTCHOME>/resource/xml/RTI_REAL_TIME_CONNECT.xml**, has the following contents:

```
<timesten_connection>
  <dsn>Example</dsn>
  <user_name>Student</user_name>
</timesten_connection>
```

Next, start *Database Integration Service* for Oracle TimesTen from the command prompt.

```
>./rtirtc_timesten -noDaemon -cfgName default
RTI Database Integration Service for Oracle, Release 5.x.y: startup suc-
ceeded
```

Database Integration Service for Oracle is now connected to the **Example** data source.

6.7 Creating an Oracle Table

Connect to the **Student** Oracle account and create a table that will be cached by Oracle TimesTen:

```
> sqlplus student/mypsswrd@orcl
SQL> set autocommit on;[1]
SQL> drop table Example;[2]
SQL> create table Example (pkey NUMBER(5), message CHAR(13), primary
key(pkey));
```

Grant SELECT, INSERT, UPDATE and DELETE privileges on the Example table to the cache administration user:

```
SQL> GRANT SELECT ON Example TO cacheuser;
SQL> GRANT INSERT ON Example TO cacheuser;
```

1. By default, *sqlplus* is not in autocommit mode. Use this statement to turn on autocommit.
2. You only need to execute this step if the table “Example” already exists because you worked through the exercises in Chapter 4: Database Integration Service for Oracle 11g on UNIX-based Systems or Chapter 5: Database Integration Service for Oracle 11g on Windows Systems.

```
SQL> GRANT UPDATE ON Example TO cacheuser;
SQL> GRANT DELETE ON Example TO cacheuser;
SQL> quit;
```

6.8 Creating a Cache Group and Start the Replication Agent

Use the *ttlsql* utility to connect to the **Example** data store. At the command prompt, call the **ttCacheStart** procedure to start the cache agent for the data store:

```
> ttlsql -connStr "DSN=Example"
Command> call ttCacheStart();
```

Next, use the CREATE CACHE GROUP statement to create an asynchronous writethrough cache group named **ExampleCache**, to cache the contents of the Example Oracle table on TimesTen:

```
Command> CREATE ASYNCHRONOUS WRITETHROUGH CACHE GROUP ExampleCache
FROM EXAMPLE (pkey TT_SMALLINT, message CHAR(13), primary key(pkey));
```

The cache group specification must include a TimesTen definition of the table(s) to be cached in Oracle TimesTen. The mapping between Oracle types and TimesTen types in the FROM clause is described in Chapter 5 of the *Database Integration Service User's Manual*.

Start the TimesTen replication agent.

```
Command> call ttRepStart();
Command> quit;
```

6.9 Storing Samples from a Publisher into the TimesTen Table

Using *ttlsql*, insert the following entry into the **RTIDDS_SUBSCRIPTIONS** table in the TimesTen data store to capture the samples from the **MyType_publisher** application. This is the same application that you created and compiled in either Storing Samples from Publishers (Section 2.7) or Storing Samples from Publishers (Section 3.8).

```
> ttlsql -connStr "DSN=Example"
Command> insert into RTIDDS_SUBSCRIPTIONS values
('Student', 'Example', 0, 'Example MyType', 'MyType');
Command> quit;
```

Start the **MyType_publisher** application from the command prompt.

```
> objs/<arch>/MyType_publisher
Writing MyType, count 0
Writing MyType, count 1
Writing MyType, count 2
```

The published samples are now being stored in the TimesTen **Example** table. These changes are automatically being synchronized to the Oracle **Example** table by Cache Connect to Oracle as configured by the cache group **ExampleCache**.

Check that this is happening by examining the contents of the Oracle table using **SQL*Plus**:

```
> sqlplus student/mypsswrd@orcl
SQL> set autocommit on;
SQL> select * from Example;
```

The output will look something like this:

```
PKEY MESSAGE       RTIDDS_DOMAIN_ID RTIRTC_REMOTE
```

```
---------- ------------- ---------------- -------------
        90 Hello world!                44             1
        91 Hello world!                44             1
        92 Hello world!                44             1
```

The actual number of rows found depends on when the *Database Integration Service* Daemon started storing samples. If you execute the **select** statement repeatedly, you will see the number of rows grow as new values are published by **MyType_publisher** every 4 seconds and the changes are stored into TimesTen by the *Database Integration Service* Daemon, then replicated to Oracle with Oracle In-Memory Database Cache.

This example shows how *Database Integration Service* can be used in a system where real-time applications produce data at high rates that need to be stored in a persistent database. In this solution, *Database Integration Service* for Oracle uses the Oracle TimesTen In-Memory database as a high-performance data cache and Oracle In-Memory Database Cache to record the data permanently in an Oracle database.

6.10 Shutting Down

When you are finished with this example, you should shut down Oracle with Oracle In-Memory Database Cache (or else it will continue to run in the background).

1. Stop the replication agent:

```
> ttlsql -connStr "DSN=Example"
Command> call ttRepStop();
```

2. Drop the cache group:

```
Command> DROP CACHE GROUP ExampleCache;
```

3. Stop the cache agent:

```
Command> call ttCacheStop();
Command> quit;
```

Chapter 7 Database Integration Service for MySQL on UNIX-based Systems

7.1 Before Installing

Before you can run *Database Integration Service*:

- **Verify that MySQL is installed and running on your system. The required version of MySQL is listed in the *Release Notes*.**

  The installation of MySQL is beyond the scope of this document. Please see the MySQL Reference Manual for the process to install and configure MySQL.

- You also need the MySQL ODBC driver. See the *Release Notes* for the required version. The driver is not bundled with the MySQL server and must be installed separately.

  The ODBC connector can be downloaded from http://dev.mysql.com/downloads/connector/odbc/5.1.html.

  The installation guide can be found in http://dev.mysql.com/doc/refman/5.1/en/connector-odbc-installation.html.

- The MySQL ODBC driver requires an ODBC driver manager. We recommend using UnixODBC 2.2.12 (or higher), a complete, free/open ODBC solution for UNIX-based systems. The driver manager can be downloaded from http://www.unixodbc.org.

- **Note for UnixODBC:** *Database Integration Service* links to the UnixODBC library **libodbc.so.1**. In release 2.3.1, UnixODBC changed the library version from 1 to 2. If, after installing UnixODBC, *Database Integration Service* cannot find **libodbc.so**, create a symlink to **libodbc.so.1** from **libodbc.so.2**:

  ```
  > ln -s libodbc.so.2 libodbc.so.1
  ```

7.2 Installation

To install Database Integration Service:

1. Install *Database Integration Service* on top of *RTI Connext DDS*. There are two ways to install it, from *RTI Launcher* or from the command line.

   <NDDSHOME> is described in Paths Mentioned in Documentation (Section 1.1).

To install from RTI Launcher:

a. Start *RTI Launcher*:

```
cd <NDDSHOME>/bin
./rtilauncher
```

b. From the Utilities tab, click on **RTI Package Installer**.

c. Use the + sign to add the **.rtipkg** file that you want to install.

d. Click **Install**.

To install from the command line:

```
cd <NDDSHOME>/bin
./rtipkginstall <path to .rtipkg file>
```

2. Add the path to the UnixODBC driver manager to the beginning of **LD_LIBRARY_PATH**. For example:

```
> setenv LD_LIBRARY_PATH /usr/lib:$LD_LIBRARY_PATH
```

or

```
> export LD_LIBRARY_PATH=/usr/lib:$LD_LIBRARY_PATH
```

Replace **/usr/lib** with the location of the UnixODBC driver manager in your system.

7.3 Configuring the MySQL Server

The *Database Integration Service* Daemon uses User-Defined Functions (UDFs) to interface with the MySQL server. Or more accurately, *Database Integration Service* provides UDFs in a library, **librtirtc_mysqlq.so** (under the **lib/*<arch>*** directory). The MySQL server must be able to load this library while running to communicate with the *Database Integration Service* Daemon.

This section provides instructions on how to install **librtirtc_mysqlq.so**.

7.3.1 Install librtirtc_mysqlq.so

Copy the appropriate version of **librtirtc_mysqlq.so** into the MySQL server's plugin directory (the directory named by the **plugin_dir** system variable).

To check the current location of the plugin directory, login to MySQL and execute the following statement:

```
> mysql -uroot (you can log in as any user)
mysql> show variables like 'plugin_dir';
+---------------+----------------------------------------+
| Variable_name | Value                                  |
+---------------+----------------------------------------+
| plugin_dir    | /opt/mysql/product/5.1.44/lib/plugin |
+---------------+----------------------------------------+
1 row in set (0.14 sec)
```

The plugin directory can be changed by setting the value of **plugin_dir** when the MySQL server is started. For example, you can set its value in the **my.cnf** configuration file:

```
[mysqld]
plugin_dir=/path/to/plugin/directory
```

For more information about the plugin directory see http://dev.mysql.com/doc/refman/5.1/en/install-plugin.html.

7.3.2 Install libnddsc.so and libnddscore.so

This section only applies to Linux systems.

librtirtc_mysqlq.so uses *Connext DDS* and thus the shared libraries **libnddsc.so** and **libnddscore.so** must also be installed and be accessible at run time by the MySQL server. For that purpose, also make sure the permissions of the folder containing the **nddsc.dll** and **nddscore.dll** libraries are appropriate for the user who is running the MySQL server.

Add the directory containing the appropriate *Connext DDS* libraries to the environment variable, **LD_LIBRARY_PATH**, for the user who starts the MySQL server. You may need to restart the MySQL server after this variable has been changed.

Note: If you are using a license managed version of RTI Connext DDS (e.g., an evaluation installer), make sure that you define the RTI_LICENSE_FILE environment variable to point to a valid license file. Also make sure that this environment variable is visible from the MySQL server process. In order to take effect, on some systems you may need to define it as a system-wide environment variable and restart the MySQL database process. You can find more information about using a license file in Section 2.4 of the *RTI Connext DDS Core Libraries Getting Started Guide*.

7.4 Creating a MySQL Account

Before you can use *Database Integration Service*, you need to obtain a MySQL user account from your database administrator. If you are acting as your own database administrator, start **mysql** from the command prompt to connect to the MySQL server as the MySQL **root** user:

```
> mysql -uroot
```

If you have assigned a password to the **root** account, you will also need to provide the **-p** option.

For example, to create a new MySQL account with user name **Student** and password **mypsswrd**, enter the following:

```
> mysql -uroot
mysql> GRANT ALL PRIVILEGES ON *.* TO 'Student'@'localhost' IDENTIFIED BY 'mypss-
wrd' WITH GRANT OPTION;
```

The remaining sections in this chapter assume that a MySQL user named **Student** with the password **mypsswrd** has an account on the local host.

7.5 Creating a Data Source for MySQL

Database Integration Service uses the MySQL ODBC driver to access data sources. Usually these are the same data sources to which your applications connect. The connection information for each data source is stored in the **.odbc.ini** file. The stored information describes each data source in detail, specifying the driver name, a description, and any additional information the driver needs to connect to the data source.

To create the **.odbc.ini** file, follow these steps:

1. Create a new file named **.odbc.ini** in your home directory using your favorite text editor. Alternatively, you can use the **ODBCINI** environment variable to specify the file location.
2. Insert these lines in the file:

```
[ODBC Data Source]
Example=MySQL Driver
```

```
[Example]
DRIVER=/usr/lib/libmyodbc5.so
Database=test
```

Notes:

- Make sure that **DRIVER** points to the valid location of the MySQL ODBC driver on your system.
- When connecting to a MySQL server located on the local system, the mysql client connects through a local file called a socket instead of connecting to the localhost loopback address 127.0.0.1. For the mysql client, the default location of this socket file is **/tmp/mysql.sock**. However, many MySQL installations place this socket file somewhere else, such as **/var/lib/mysql/mysql.sock**. You may see this error message when you start the daemon:

  ```
  [unixODBC][MySQL][ODBC 3.51 Driver]Can't connect to
  local MySQL server through socket '/tmp/mysql.sock'
  ```

  To correct this error, specify the right socket file by adding the SOCKET attribute to the DSN entry. For example:

  ```
  [Example]
  DRIVER=/usr/lib/libmyodbc3.so
  SOCKET=/var/lib/mysql/mysql.sock
  Database=test
  ```

3. Save your changes.

The “Example” data source uses the “test” database which is usually available as a workspace for users to try things out.

7.6 Creating a Configuration File for Database Integration Service

Database Integration Service reads its configuration information from a file. By default, *Database Integration Service* tries to load the configuration file, **<*Database Integration Service executable location*>/../../resource/xml/RTI_REAL_TIME_CONNECT.xml**. You can specify a different file with the command-line option, **-cfgFile**.

The default file, **RTI_REAL_TIME_CONNECT.xml**, does not contain any actual valid information yet. For this example we will edit this file as follows:

1. Look for the tag **<mysql_connection>**. Replace the tags **<dsn>, <user_name>**, and **<password>** as follows:

```
<mysql_connection>
  <dsn>Example</dsn>
  <user_name>Student</user_name>
  <password>mypsswrd</password>
</mysql_connection>
```

2. Save the file.

This configuration file instructs *Database Integration Service* to monitor the data source as specified by the "Example" DSN.

7.7 Running the MySQL Server in ANSI_QUOTES Mode

Database Integration Service requires the MySQL server to be configured in ANSI_QUOTES mode. Under that configuration, the MySQL server treats “” as an identifier quote character and not as a string quote character.

To verify if the MySQL server is already configured in ANSI_QUOTES mode, run the following SQL statement from the **mysql** command prompt:

```
mysql> SELECT @@global.sql_mode;
```

If the string ‘ANSI_QUOTES’ is not part of the result, the MySQL server needs to be configured in ANSI_QUOTES mode by executing the following SQL statement:

```
mysql> SET GLOBAL sql_mode = 'ANSI_QUOTES';
```

Note: If you want to set permanently the sql_mode, you may want to configure it in the MySQL Server configuration file. Otherwise, you will need to set the sql_mode every time you restart the MySQL server.

7.8 Starting the Database Integration Service Daemon

Start *Database Integration Service* by executing the following commands.

<NDDSHOME> is described in Paths Mentioned in Documentation (Section 1.1).

```
> cd <NDDSHOME>/bin
> ./rtirtc_mysql -noDaemon -cfgName default
```

By default, *Database Integration Service* runs in the background as a daemon process. Specifying the “**-noDaemon**” option prevents that, and starts up the *Database Integration Service* Daemon as a regular process. Messages are sent to standard output.

You should see the following output, indicating that the process is running.

```
>./rtirtc_mysql -noDaemon -cfgName default
RTI Database Integration Service for MySQL, Release 5.x.y: startup succeeded
```

Database Integration Service is now connected to the “**Example**” data source.

Note: Make sure that you have the UnixODBC driver manager library, **libodbc.so**, on your **LD_LIBRARY_PATH (**see Step 2 in Before Installing (Section 7.1)).

7.9 Storing Samples from Publishers

In this section, you will enable automatic capturing of data samples in MySQL database. The first step is to create an IDL type definition. By using the “**-example**” option of *rtiddsgen*[1] you will automatically create a Publisher of this IDL type.

1. Create a new text file called “**MyType.idl**” with the following contents:

```
struct MyType {
    short pkey; //@key
    char message[13];
};
```

1. Please see the *RTI Code Generator User’s Manual* for more information about how to run *rtiddsgen*.

This IDL file specifies a data type that contains a message. Each instance is uniquely identified by the "**pkey**" field.

2. Now execute the following command to compile the IDL type.

   <NDDSHOME> is described in Paths Mentioned in Documentation (Section 1.1).

   ```
   > <NDDSHOME>/bin/rtiddsgen -language C -example <arch> MyType.idl
   ```

   For example:

   ```
   > <NDDSHOME>/bin/rtiddsgen -language C -example i86Linux2.6gcc4.1.1 MyType.idl
   ```

   This generates the **MyType**, **MyTypePlugin**, and **MyTypeSupport** files, as well as the **MyType_publisher** and **MyType_subscriber** example code.

3. The generated example will also have a makefile named:

   ```
   makefile_MyType_<arch>
   ```

   You may need to edit the makefile to specify the location of the compiler if it is not available on your path.

4. Edit **MyType_publisher.c**, and find the line containing the comment:

   ```
   /* Modify the data to be written here */
   ```

   Insert the following lines immediately below this comment:

   ```
   instance->pkey = count;
   strcpy(instance->message, "Hello world!");
   ```

5. Save your changes and build the **MyType_publisher** and **MyType_subscriber** applications by executing:

   ```
   > gmake -f makefile_MyType_<arch>
   ```

6. Start the **MyType_publisher** application so that it starts publishing data samples.

   ```
   > objs/<arch>/MyType_publisher
   ```

   On the screen, you will see:

   ```
   Writing MyType, count 0
   Writing MyType, count 1
   Writing MyType, count 2
   ...
   ```

   The samples are not captured in the MySQL database yet. For this you need to set up a subscription in *Database Integration Service*.

   Subscriptions are set up in the "**RTIDDS_SUBSCRIPTIONS**" configuration table that *Database Integration Service* created when it connected to the MySQL database.

   Start **mysql** from the command prompt:

   ```
   > mysql -uStudent -pmypsswrd test
   mysql>
   ```

7. You can see that the "**RTIDDS_SUBSCRIPTIONS**" configuration table is still empty at this point by executing the following SQL command—*don't forget to type a semicolon ';' at the end of the line:*

   ```
   mysql> select * from RTIDDS_SUBSCRIPTIONS;
   Empty set (0.01 sec)
   ```

8. To store the samples from the **MyType_publisher** application in a table named "**Example**", insert a corresponding entry into the "**RTIDDS_SUBSCRIP-TIONS**" table:

```
mysql> insert into RTIDDS_SUBSCRIPTIONS (table_owner, table_name, domain_id, top-
ic_name, type_name) values
('test', 'Example', 0, 'Example MyType', 'MyType');
1 row inserted.
```

   This entry directs *Database Integration Service* to create a user table named "**Example**" and to start storing samples published with topic "**Example MyType**" and data type "**MyType**" in Domain **0**.

   Note: The **table_owner** in MySQL is the database.

9. If the **MyType_publisher** application is still running, you can execute the following SQL statement to view the contents of the table—otherwise, restart **MyType_publisher** as described above before executing this statement.

```
mysql> select * from Example;
```

   The output will look something like this:

```
+------+-------------+------------------+--------------+------------+
| pkey | message     | RTIDDS_DOMAIN_ID | RTIRTC_REMOTE | RTIRTC_SCN |
+------+-------------+------------------+--------------+------------+
...
|  134 | Hello world! |                0 |            1 |          0 |
|  135 | Hello world! |                0 |            1 |          0 |
|  136 | Hello world! |                0 |            1 |          0 |
+------+-------------+------------------+--------------+------------+
85 rows in set (0.00 sec)
```

   The actual number of rows found depends on exactly when the *Database Integration Service* Daemon started storing samples. If you execute the "**select**" statement repeatedly, you will see the number of rows grow. This is because the **MyType_publisher** application writes a new instance every 4 seconds.

7.10 Publishing Database Updates to Subscribers

In this section, you will enable *Database Integration Service* to automatically publish changes made to a table in a data source.

We assume that you have followed the instructions in Starting the Database Integration Service Daemon (Section 7.8) to start a *Database Integration Service* Daemon. In addition, we assume that you have followed the instructions in Steps 1 to 4 in Storing Samples from Publishers (Section 7.9) to create the IDL file and generate and compile the example code.

1. If you have not already done so, stop the **MyType_publisher** application by pressing "**Ctrl-c**" in the window where it is running.

2. Start the **MyType_subscriber** application from the command line:

```
> objs/<arch>/MyType_subscriber
MyType subscriber sleeping for 4 sec...
MyType subscriber sleeping for 4 sec...
...
```

   You will get this message repeatedly, since nothing is being published.

3. For publishing changes to a data source, you need to configure the "**RTIDDS_PUBLICATIONS**" table. For publishing the changes to the "**Example**" table, execute the following SQL statement:

```
mysql> insert into RTIDDS_PUBLICATIONS (table_owner, table_name, domain_id, top-
ic_name, type_name) values
('test', 'Example', 0, 'Example MyType', 'MyType');
```

   This entry directs *Database Integration Service* to start publishing changes to table "**Example**" as the topic "**Example MyType**" with type "**MyType**" in Domain **0**.

4. Now change one of the previously captured samples in the "**Example**" table, such as the last one:

```
mysql> update Example set message = 'Hello again!' where pkey = (select * from
(select max(pkey) from Example) as _max);
```

   You will see that the **MyType_subscriber** application reports the update, for example:

```
MyType subscriber sleeping for 4 sec...
pkey: 748
   message:
      message[ 0]: 'H'
      message[ 1]: 'e'
      message[ 2]: 'l'
      message[ 3]: 'l'
      message[ 4]: 'o'
      message[ 5]: ' '
      message[ 6]: 'a'
      message[ 7]: 'g'
      message[ 8]: 'a'
      message[ 9]: 'i'
      message[10]: 'n'
      message[11]: '!'
      message[12]: <0>
MyType subscriber sleeping for 4 sec...
...
```

5. You can also update all entries in the table:

```
mysql> update Example set message = 'Hello again!';
```

   Notice that the **MyType_subscriber** application receives all changes, possibly a large number.

Chapter 8 Database Integration Service for MySQL on Windows Systems

This chapter provides instructions on how to install *Database Integration Service* for MySQL on Windows platforms.

This chapter assumes you will be *compiling* with Microsoft Visual Studio 2013[1].

8.1 Before Installation

Before you can run *Database Integration Service*, you will also need:

- **MySQL ODBC 5.1.6 driver or higher. The driver is not bundled with the MySQL server and must be installed separately.**

  The ODBC connector can be downloaded from:
  http://dev.mysql.com/downloads/connector/odbc/5.1.html

  The installation guide can be found here:
  http://dev.mysql.com/doc/refman/5.1/en/connector-odbc-installation.html

- Visual C++ Redistributable for Visual Studio 2010.

  It is available from this Microsoft website:
  http://www.microsoft.com/en-ca/download/details.aspx?id=30679

- MySQL (must be installed and running).

  See the *Release Notes* for the correct version of MySQL for your platform. The installation of MySQL is beyond the scope of this document. Please see the MySQL Reference Manual for how to install and configure MySQL.

8.2 Installation

Install *Database Integration Service* on top of *RTI Connext DDS*. There are two ways to install it, from *RTI Launcher* or the command line.

<NDDSHOME> is described in Paths Mentioned in Documentation (Section 1.1)

1. You may use other supported Microsoft compilers such as Visual Studio 2010 or Visual Studio 2015, however, the instructions in this chapter were written assuming Visual Studio 2013.

From RTI Launcher:

1. Start *RTI Launcher* from the Start menu or from the command line:

```
cd <NDDSHOME>\bin
rtilauncher
```

2. From the Utilities tab, click on **RTI Package Installer**.
3. Use the + sign to add the **.rtipkg** file that you want to install.
4. Click **Install**.

From the command line:

```
cd <NDDSHOME>\bin
rtipkginstall <path to .rtipkg file>
```

8.3 Configuring the MySQL Server

The *Database Integration Service* Daemon uses user-defined functions, UDFs, to interface with the MySQL server. Or more accurately, *Database Integration Service* provides UDFs in a library, **rtirtc_mysqlq.dll** (in the **lib\<*arch*>** directory), that the MySQL server must be able to load while running to communicate with the *Database Integration Service* Daemon.

This section provides instructions on how to install **rtirtc_mysqlq.dll**.

8.3.1 Installing rtirtc_mysqlq.dll

Copy **<NDDSHOME>\lib\<*arch*>\rtirtc_mysqlq.dll** into the MySQL server's plugin directory (the directory named by the **plugin_dir** system variable). To check the current location of the plugin directory, login to MySQL and execute the following statement:

```
> mysql -uroot (you can log in as any user)

mysql> show variables like 'plugin_dir';
```

Variable_name	Value
plugin_dir	C:\Program Files\MySQL\MySQL Server 5.1\lib/plugin

```
1 row in set (0.03 sec)
```

The plugin directory can be changed by setting the value of **plugin_dir** when the MySQL server is started. For example, you can set its value in the **my.cnf** configuration file:

```
[mysqld]
plugin_dir=/path/to/plugin/directory
```

For additional information about the plugin directory, see the following link:
http://dev.mysql.com/doc/refman/5.1/en/install-plugin.html.

8.3.2 Installing nddsc.dll and nddscore.dll

Because **rtirtc_mysqlq.dll** uses *Connext DDS*, the dynamic libraries **nddsc.dll** and **nddscore.dll** must also be installed and accessible at runtime by the MySQL server.

Make sure your **Path** system environment variable[1] contains the path to **nddsc.dll** and **nddscore.dll** (in **<NDDSHOME>\lib\<*arch*>**).

If **<NDDSHOME>\lib\<*arch*>** is not already in your **Path**, you will need to restart your computer after you modify the **Path**. If you do not want to restart your computer, you can copy the libraries into a directory that is already in the system **Path**, such as **c:\Windows\System32**.

Note: If you are using a license managed version of RTI Connext DDS (e.g., an evaluation installer), make sure that you define the RTI_LICENSE_FILE environment variable to point to a valid license file. Also make sure that this environment variable is visible from the MySQL server process. In order to take effect, on some systems you may need to define it as a system-wide environment variable and restart the MySQL database process. You can find more information about using a license file in Section 2.4 of the *RTI Connext DDS Core Libraries Getting Started Guide*.

8.4 Creating a MySQL Account

Before you can use *Database Integration Service*, you need to obtain a MySQL user account from your database administrator. If you are acting as your own database administrator, start **mysql** from the command prompt to connect to the MySQL server as the MySQL **root** user:

```
> mysql -uroot
```

If you have assigned a password to the **root** account, you will also need to provide a **-p** option.

For example, to create a new MySQL account with a user name of "**Student**" and a password of "**mypsswrd**", enter the following (all on one line):

```
mysql> GRANT ALL PRIVILEGES ON *.* TO 'Student'@'localhost'
       IDENTIFIED BY 'mypsswrd' WITH GRANT OPTION;
```

The remaining sections in this chapter assume that a MySQL user named "**Student**" with the password "**mypsswrd**" has an account on the local host.

8.5 Creating a Data Source for MySQL

Database Integration Service uses the MySQL ODBC driver to access data sources. Usually these are the same data sources to which your applications connect. The connection information for each data source is stored in the Windows registry. The stored information describes each data source in detail, specifying the driver name, a description, and any additional information the driver needs to connect to the data source.

To add a data source, follow these steps:

1. Open the ODBC Data Source Administrator:
   - Select **Start, Control Panel, System and Security, Administrative Tools, Data Sources (ODBC).**
2. Select the **User DSN** tab.
3. Click the **Add** button; the **Create New Data Source** dialog appears.
4. Select the **MySQL** driver from the list of drivers.

   Most recent versions of the MySQL ODBC connector include ANSI and Unicode ODBC drivers. In such cases, use the ANSI driver for interacting with *Database Integration Service*.

1. To view and/or edit the **Path** system environment variable, from the **Start** button/menu, select **Settings, Control Panel, System, Advanced tab,** then select the **Environment Variable** button. View or edit the **Path** in the System Variables. (The exact steps for accessing environment variables may vary, depending on your version of the Windows operating system).

5. Click the **Finish** button; the 'MySQL Connector/ODBC Data Source Configuration' dialog will appear.
6. Fill out the fields in the dialog.
   a. Enter "**Example**" as the **Data Source Name** (DSN).
   b. Enter "**Student**" as the **User** and "**mypsswrd**" as the **Password**.
   c. Select "**test**" as the **Database**.
   d. All other fields can be left empty.
7. Click the **OK** button.

8.6 Creating a Configuration File for Database Integration Service

Database Integration Service reads its configuration information from a file. By default, *Database Integration Service* tries to load the configuration file, **<NDDSHOME>\resource\xml\RTI_REAL_TIME_CONNECT.xml**. You can specify a different file using the command line option, **-cfgFile**.

The default file **RTI_REAL_TIME_CONNECT.xml** does not contain any actual configuration information yet. For this example we will edit this file as follows:

1. Look for the tag **<mysql_connection>**. Replace the tags **<dsn>**, **<user_name>,** and **<password>** as follows:

```
<mysql_connection>
  <dsn>Example</dsn>
  <user_name>Student</user_name>
  <password>mypsswrd</password>
</mysql_connection>
```

2. Save the file.

This configuration file instructs *Database Integration Service* to monitor the data source as specified by the "Example" DSN.

8.7 Using Database Integration Service as a Windows Service

8.7.1 Enabling Database Integration Service to Run as a Windows Service

If you want to run *Database Integration Service* for MySQL as a Windows Service, you must install it as such before running it. To install it as a Windows Service, run the following command in a terminal with Administrator privileges:

```
<NDDSHOME>\bin\rtirtc_mysql -installService
```

8.7.2 Running Database Integration Service as a Windows Service

Windows Services automatically run in the background when the system reboots.

To run *Database Integration Service* for MySQL as a Windows Service from the command line, use the Windows **sc** utility:

```
sc rtirtc_mysql530 start
```

Or you can run the application directly without the **-noDaemon** argument:

```
<NDDSHOME>\bin\rtirtc_mysql
```

To stop *Database Integration Service* for MySQL when it is running as a Windows Service, use this command:

```
sc rtirtc_mysql530 stop
```

Alternatively, you can start/stop *Database Integration Service* for MySQL from the Windows Services Control Manager. From the **Start** menu, select **Control Panel**, **Administrative Services**, **Services**. Click on the service in the list, then right-click to select **Start** or **Stop**.

8.7.3 Disabling Database Integration Service from Running as a Windows Service

To remove *Database Integration Service* for MySQL from the list of Windows Services on your system, run this command in a terminal with Administrator privileges:

```
<NDDSHOME>\bin\rtirtc_mysql -uninstallService
```

8.8 Running the MySQL Server in ANSI_QUOTES Mode

Database Integration Service requires the MySQL server to be configured in ANSI_QUOTES mode. Under that configuration, the MySQL server treats '"' as an identifier quote character and not as a string quote character.

To verify if the MySQL server is already configured in ANSI_QUOTES mode, run the following SQL statement from the **mysql** command prompt:

```
mysql> SELECT @@global.sql_mode;
```

If the string 'ANSI_QUOTES' is not part of the result, the MySQL server needs to be configured in ANSI_QUOTES mode by executing the following SQL statement:

```
mysql> SET GLOBAL sql_mode = 'ANSI_QUOTES';
```

Note: If you want to set permanently the sql_mode, you may want to configure it in the MySQL Server configuration file. Otherwise, you will need to set the sql_mode every time you restart the MySQL server.

8.9 Starting the Database Integration Service Daemon

You can start the *Database Integration Service* for Oracle Daemon as a Windows service (assuming you followed the steps in Using Database Integration Service as a Windows Service (Section 8.7)). However, for the following example, we will start the daemon manually.

Start *Database Integration Service* by executing the following command from **<NDDSHOME>\bin**.

```
rtirtc_mysql -noDaemon -cfgName default -dbtransport 1
```

By default, *Database Integration Service* runs in the background as Windows service. Specifying the **-noDaemon** option prevents that, and starts up the *Database Integration Service* Daemon as a regular process. Messages are sent to standard output.

```
rtirtc_mysql -noDaemon -cfgName default
```

You should see the following output, indicating that the process is running.

```
RTI Database Integration Service for MySQL, Release 5.x.y: startup succeeded
```

Database Integration Service for MySQL is now connected to the "**Example**" data source.

If you see the following error message, verify that the *Database Integration Service* library **rtirtc_mysqlq.dll** is in your **Path**:

```
[DDSQLDaemonCNAHelper_createCNAConnection,line
761:ERROR:4096:50002] [CNA:CNAOpe n] Error initializing MySQL Server
Queue: [MySQL][ODBC 5.1.6 Driver][mysqld-5.1.44-community-nt]
Can't open shared library 'rtirtc_mysqlq.dll' (errno: 2)
```

If you see the following error message, verify that the *Connext DDS* libraries **nddsc.dll** and **nddscore.dll** are in your **Path** and restart the MySQL database server:

```
[DDSQLDaemonCNAHelper_createCNAConnection,line
761:ERROR:4096:50002] [CNA:CNAOpen] Error initializing MySQL Server
Queue: [MySQL][ODBC 5.1.6 Driver][mysqld-5.1.44-community-nt]
FUNCTION test.MySqlNddsQueue_initialize does not exist
```

8.10 Storing Samples from Publishers

In this section, you will enable automatic capturing of data samples in a MySQL database. The first step is to create an IDL type definition. By using the **-example** option of *rtiddsgen*[1] you will automatically create a Publisher of this IDL type.

1. Create a new file called "**MyType.idl**" with the following contents:

```
struct MyType {
    short pkey; //@key
    char message[13];
};
```

   This IDL file specifies a data type that contains a message. Each instance is uniquely identified by the "**pkey**" field.

2. Now execute the following command to compile the IDL type.

```
<NDDSHOME>\bin\rtiddsgen -language C -example i86Win32VS2013[2]
-ppDisable MyType.idl
```

   This generates the **MyType**, **MyTypePlugin**, and **MyTypeSupport** files, as well as the **MyType_publisher** and **MyType_subscriber** example code.

3. The generated example will also have a Visual Studio Solution file called **MyType-vs2013.sln**. Start Microsoft Visual Studio 2013 and load this workspace by clicking on this file.

4. Edit **MyType_publisher.c**, and find the line containing the comment:

```
/* Modify the data to be written here */
```

   Insert the following lines immediately below this comment:

```
instance->pkey = count;
strcpy(instance->message, "Hello world!");
```

5. Save your changes and build the **MyType_publisher** project in Visual Studio.

1. *rtiddsgen* is an IDL code generator distributed with *Connext DDS*. Please refer to the *RTI Code Generator User's Manual* for more information about how to run *rtiddsgen*.
2. If you are using a different supported compiler, you will need to use a different value here, such as i86Win32VS2008 for Visual Studio 2008.

6. Start the **MyType_publisher** application so that it starts publishing data samples. From a command prompt, enter:

```
> objs\i86Win32VS2013\MyType_publisher
```

   On the screen, you will see:

```
Writing MyType, count 0
Writing MyType, count 1
Writing MyType, count 2
...
```

   The samples are not captured in the MySQL database yet. For this you need to set up a subscription in *Database Integration Service*.

   Subscriptions are set up in the "**RTIDDS_SUBSCRIPTIONS**" configuration table that *Database Integration Service* created when it connected to the MySQL database.

7. Start **mysql** from the command prompt:

```
> mysql -uStudent -pmypsswrd test
mysql>
```

8. You can see that the "**RTIDDS_SUBSCRIPTIONS**" configuration table is still empty at this point by executing the following SQL command—*don't forget to type a semicolon ';' at the end of the line*:

```
mysql> use test;
mysql> select * from RTIDDS_SUBSCRIPTIONS;
Empty set (0.01 sec)
```

9. To store the samples from the **MyType_publisher** application in a table named "**Example**", insert a corresponding entry into the "**RTIDDS_SUBSCRIPTIONS**" table:

```
mysql> insert into RTIDDS_SUBSCRIPTIONS (table_owner, table_name,
domain_id, topic_name, type_name) values
('test', 'Example', 0, 'Example MyType', 'MyType');
1 row inserted.
```

   This entry directs *Database Integration Service* to create a user table named "**Example**" and to start storing samples published with topic "**Example MyType**" and data type "**MyType**" in Domain **0**.

   Note: The **table_owner** in MySQL is the database.

10. If the **MyType_publisher** application is still running, you can execute the following SQL statement to view the contents of the table—otherwise, restart **MyType_publisher** as described above before executing this statement.

```
mysql> select * from Example;
```

   The output will look something like this:

pkey	message	RTIDDS_DOMAIN_ID	RTIRTC_REMOTE	RTIRTC_SCN
. . .				
134	Hello world!	0	1	0
135	Hello world!	0	1	0
136	Hello world!	0	1	0

```
85 rows in set (0.00 sec)
```

The actual number of rows depends on when exactly the *Database Integration Service* Daemon started storing samples. If you execute the "**select**" statement repeatedly you will see the number of rows grow. This is because the **MyType_publisher** application writes a new instance every 4 seconds.

8.11 Publishing Database Updates to Subscribers

In this section, you will enable *Database Integration Service* to automatically publish changes made to a table in a data source.

We assume that you have followed the instructions in Starting the Database Integration Service Daemon (Section 8.9) to start a *Database Integration Service* Daemon. In addition, we assume that you have followed the instructions in Steps 1 to 4 in Storing Samples from Publishers (Section 8.10) in creating the IDL file and generating and compiling the example code.

1. If you have not already done so, stop the **MyType_publisher** application by pressing "**Ctrl-c**" in the window where it is running.
2. Build the **MyType_subscriber** project in Visual Studio and start the application from the command line:

```
> objs\i86Win32VS2013\MyType_subscriber
MyType subscriber sleeping for 4 sec...
MyType subscriber sleeping for 4 sec...
...
```

   You will get this message repeatedly, since nothing is being published.

3. For publishing changes to a data source, you need to configure the "**RTIDDS_PUBLICATIONS**" table. For publishing the changes to the "**Example**" table, execute the following SQL statement:

```
mysql> insert into RTIDDS_PUBLICATIONS (table_owner, table_name,
domain_id, topic_name, type_name) values
('test', 'Example', 0, 'Example MyType', 'MyType');
```

   This entry directs *Database Integration Service* to start publishing changes to table "**Example**" as the topic "**Example MyType**" with type "**MyType**" in Domain **0**.

4. Now change one of the previously captured samples in the "**Example**" table, for instance the last one:

```
mysql> update Example set message = 'Hello again!' where pkey = (select * from
(select max(pkey) from Example) as _max);
```

   You will see that the **MyType_subscriber** application reports the update, for example:

```
...
MyType subscriber sleeping for 4 sec...
pkey: 748
   message:
      message[ 0]: 'H'
      message[ 1]: 'e'
      message[ 2]: 'l'
      message[ 3]: 'l'
      message[ 4]: 'o'
      message[ 5]: ' '
      message[ 6]: 'a'
      message[ 7]: 'g'
      message[ 8]: 'a'
      message[ 9]: 'i'
```

```
      message[10]: 'n'
      message[11]: '!'
      message[12]: <0>
MyType subscriber sleeping for 4 sec...
...
```

5. You can also update all entries in the table:

```
mysql> use test;
mysql> update Example set message = 'Hello again!';
```

Notice that the **MyType_subscriber** application receives all changes, possibly a large number.

Chapter 9 Database Integration Service for Microsoft SQL Server

This chapter provides instructions on how to install and use *Database Integration Service* for Microsoft® SQL Server™ on Windows platforms.

9.1 Before Installation

Before you can run *Database Integration Service*, you will also need:

- Microsoft SQL Server 2012 SP1 or higher.

  The installation of SQL Server is beyond the scope of this document. Please see SQL Server Installation instructions[1] for how to install and configure SQL Server. *Database Integration Service* requires the SQL Server "Client Tools Connectivity" component[2] for ODBC support.

- Visual C++ Redistributable for Visual Studio 2012 Update 4.

  It is available from this Microsoft website: http://www.microsoft.com/en-ca/download/details.aspx?id=30679

- Microsoft .NET Framework 4.5 or higher.

  The installation of the .NET Framework is beyond the scope of this document. Please see Microsoft's Download Center[3] for how to install .NET.

1. Microsoft documentation for SQL Server 2012 is available here: http://msdn.microsoft.com/en-us/library/bb500469(v=sql.110).aspx
2. Client Tools Connectivity in the Installation Center is documented as "Connectivity Components": http://msdn.microsoft.com/en-us/library/ms144275(v=sql.110).aspx
3. Microsoft .NET Framework 4.5 Download Center page: http://www.microsoft.com/en-us/download/details.aspx?id=30653

9.2 Installation

Install *Database Integration Service* on top of *RTI Connext DDS*. There are two ways to install it, from *RTI Launcher* or the command line.

<NDDSHOME> is described in Paths Mentioned in Documentation (Section 1.1).

From RTI Launcher:

1. Start *RTI Launcher* from the Start menu or from the command line:

```
cd <NDDSHOME>\bin
rtilauncher
```

2. From the Utilities tab, click on **RTI Package Installer**.
3. Use the + sign to add the **.rtipkg** file that you want to install.
4. Click **Install**.

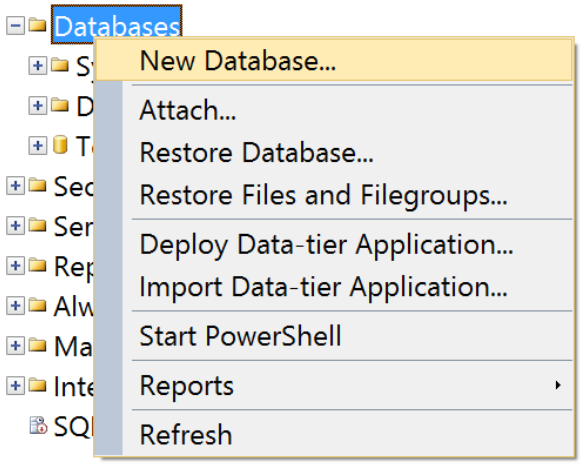
From the command line:

```
cd <NDDSHOME>\bin
rtipkginstall <path to .rtipkg file>
```

9.3 Configuring a SQL Server Login

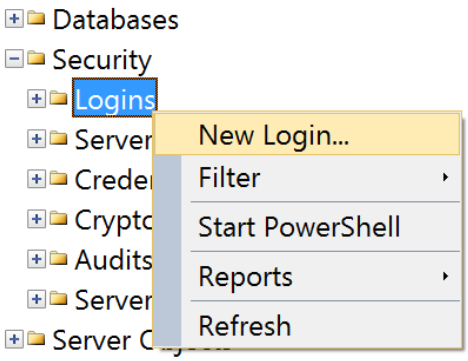
Before you can use *Database Integration Service*, you need to obtain a SQL Server login for an example database from your database administrator. If you are acting as your own database administrator, connect to the server with Microsoft SQL Server Management Studio.

1. In the Object Explorer, expand ***<Your Server Name>***. Right-click to open the context menu for **Databases** and select **New Database…**



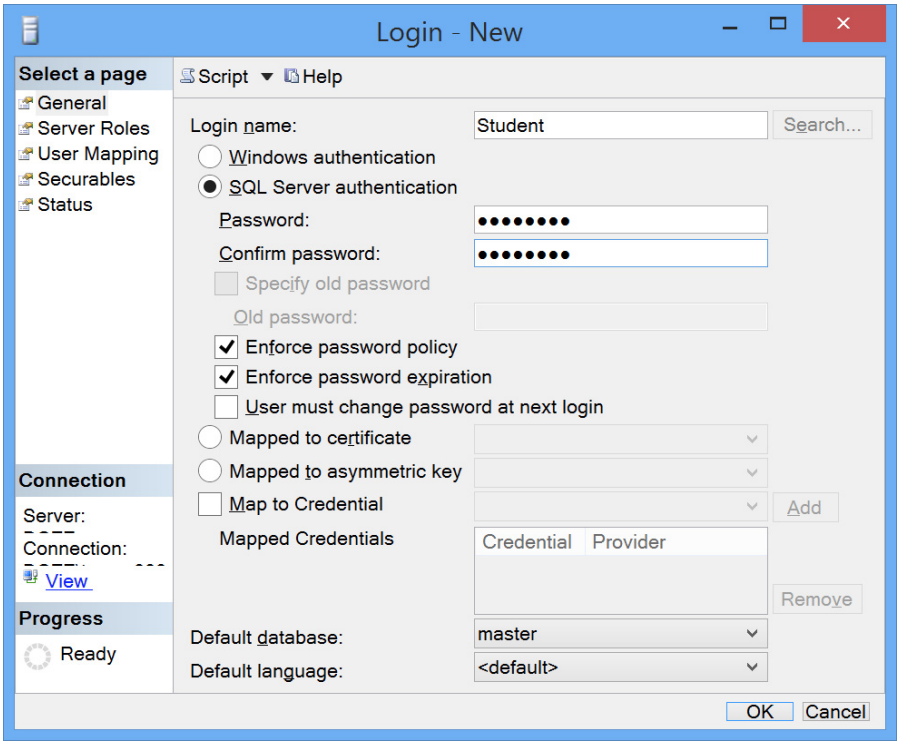
2. In the **New Database** dialog box, enter the name **Test** for the database and click **OK**.

3. In the Object Explorer, expand ***<Your Server Name>*** and **Security**. Right-click to open the context menu for **Logins** and select **New Login…**



4. If you are using Windows authentication, find the local or Domain user to add to SQL Server: enter the username in the **Login name:** box, select the radio button for **Windows authentication**, and click **OK**.

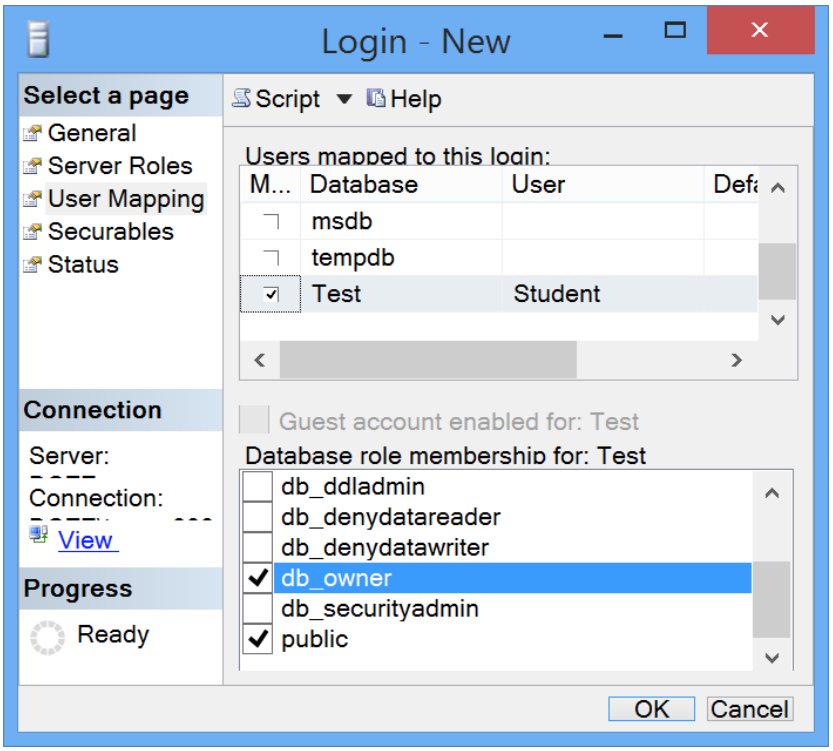
   Otherwise, select the **SQL Server authentication** radio button and enter password information. Note that the **'User must change password at next login'** option is selected by default.



The remaining sections in this chapter assume that a user named **Student** with the password **mypsswrd** has a login on the SQL Server. This login may use Windows authentication or SQL Server authentication.

9.3.1 Permissions

Database Integration Service depends on SQL Server features such as Change Tracking and the Service Broker in order to publish changes from a table to a DDS *Topic*. If the database or a monitored table do not have these enabled, *Database Integration Service* will issue **ALTER DATABASE** and **ALTER TABLE** queries to enable them. In order to grant *Database Integration Service* permission for this, simply assign the login to the **db_owner** database role for the **Test** database on the **User Mapping** page when creating the login. Your database administrator may assign permissions with a finer granularity, or may even alter the necessary objects manually. These procedures are specific to your environment and outside the scope of *Database Integration Service* documentation.



9.4 Creating a Data Source for SQL Server

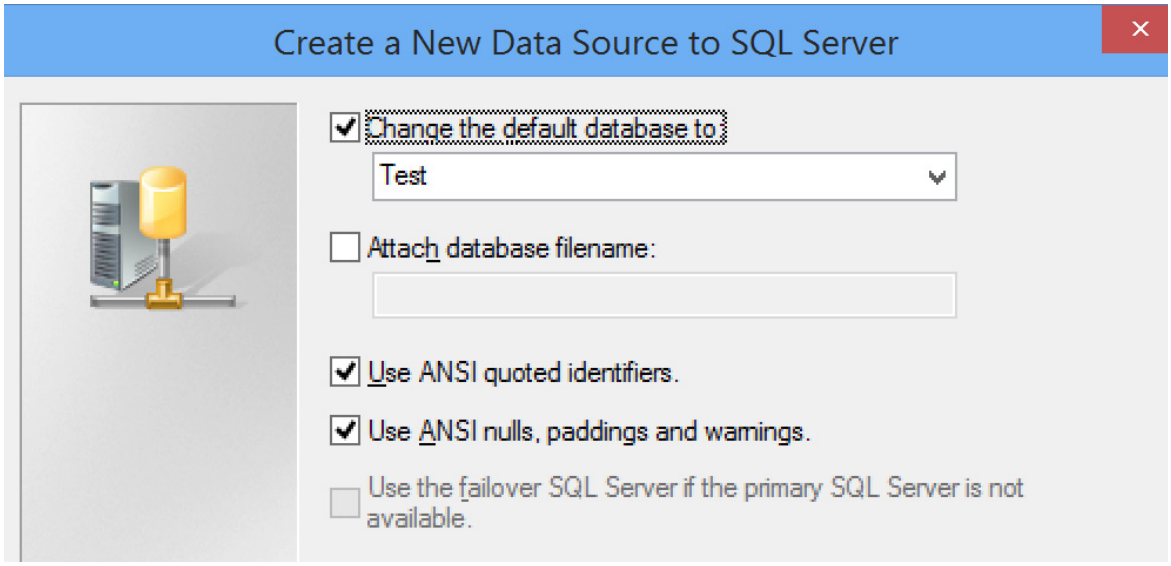
Database Integration Service uses ODBC to access data sources. Usually these are the same data sources to which your applications connect. The connection information for each data source is stored in the Windows registry. The stored information describes each data source in detail, specifying the driver name, a description, and any additional information the driver needs to connect to the data source.

To add a data source, follow these steps:

1. Open the ODBC Data Source Administrator:
   - Select **Start**, **Control Panel**, **System and Security**, **Administrative Tools, Data Sources (ODBC)**.
2. Select the **User DSN** tab.

   In this document *Database Integration Service* will be run from the command line. If you want to run the daemon as a Window service, select **System DSN** instead of User DSN.
3. Click the **Add** button; the **Create New Data Source** dialog appears.

4. Select the **SQL Server** driver from the list of drivers.
5. Click the **Finish** button; the **Create a New Data Source to SQL Server** dialog appears.
6. Fill in the fields in the dialog.
   - Enter **Example** as the **Name (DSN)**.
   - Select your SQL Server as the **Server**.
7. Click the **Next** button; the authentication options appear.
8. Choose either **Windows** authentication or **SQL Server** authentication**.** For the latter, enter **Student** as the Login ID and **mypsswrd** as the Password. Click **Next**.
9. Select **Test** from the menu labeled **Change the default database to:**.



10. Click **Next**, then **Finish**.

Database Integration Service only interacts with the database specified in the ODBC DSN.

9.5 Creating a Configuration File for Database Integration Service

Database Integration Service reads its configuration information from a file. By default, *Database Integration Service* tries to load a configuration file in the working directory called **USER_REAL_TIME_CONNECT.xml**. You can specify a different file using the command-line option **-cfgFile**. The remaining sections in this chapter assume that you created the file **C:\Student\USER_REAL_TIME_CONNECT.xml** as follows:

Windows Authentication	SQL Server Authentication
<dds> <real_time_connect name="Test"> <sqlserver_connection> <dsn>Example</dsn> </sqlserver_connection> </real_time_connect> </dds>	<dds> <real_time_connect name="Test"> <sqlserver_connection> <dsn>Example</dsn> <user_name>Student</user_name> <password>mypsswrd</password> </sqlserver_connection> </real_time_connect> </dds>

This configuration file instructs *Database Integration Service* to monitor the data source as specified by the **Example** DSN.

9.6 Using Database Integration Service as a Windows Service

9.6.1 Enabling Database Integration Service to Run as a Windows Service

If you want to run *Database Integration Service* for SQL Server as a Windows Service, you must install it as such before running it. To install it as a Windows Service, run the following command in a terminal with Administrator privileges:

```
<NDDSHOME>\bin\rtirtc_sqlserver -installService
```

9.6.2 Running Database Integration Service as a Windows Service

Windows Services automatically run in the background when the system reboots.

To run *Database Integration Service* for SQL Server as a Windows Service from the command line, use the Windows **sc** utility:

```
sc rtirtc_sqlserver530 start
```

Or you can run the application directly without the **-noDaemon** argument:

```
<NDDSHOME>\bin\rtirtc_sqlserver
```

To stop *Database Integration Service* for SQL Server when it is running as a Windows Service, use this command:

```
sc rtirtc_sqlserver530 stop
```

Alternatively, you can start/stop *Database Integration Service* for SQL Server from the Windows Services Control Manager. From the **Start** menu, select **Control Panel**, **Administrative Services**, **Services**. Click on the service in the list, then right-click to select **Start** or **Stop**.

9.6.3 Disabling Database Integration Service from Running as a Windows Service

To remove *Database Integration Service* for SQL Server from the list of Windows Services on your system, run this command in a terminal with Administrator privileges:

```
<NDDSHOME>\bin\rtirtc_sqlserver -uninstallService
```

9.7 Starting the Database Integration Service Daemon

You can start the *Database Integration Service* for Oracle Daemon as a Windows service (assuming you followed the steps in Using Database Integration Service as a Windows Service (Section 9.6)). However, for the following example, we will start the daemon manually.

Either use an absolute path to the *Database Integration Service* executable (in <NDDSHOME>\bin) or adjust your **PATH** environment variable accordingly.

Note: If you start *Database Integration Service* as a Windows service on Windows 2003 or newer platforms, the shared-memory transport is not supported. For details on how to configure DDS applications to use different transport settings, please see the *RTI Connext DDS Core Libraries User's Manual* (Section 8.5.7, TRANSPORT_BUILTIN QosPolicy).

1. Start *Database Integration Service* by executing the following commands:

```
cd Student
rtirtc_sqlserver -noDaemon -cfgName Test
```

   By default, *Database Integration Service* runs in the background as a Windows service. Specifying the **-noDaemon** option prevents that, and starts up the *Database Integration Service* Daemon as a regular process. Log messages are sent to standard output.

   You should see the following output, indicating that the process is running.

```
RTI Database Integration Service to SQL Server, Release 5.x.y: startup succeeded
```

   Database Integration Service is now connected to the **Example** data source.

9.8 Storing Samples from Publishers

In this section, you will enable automatic capturing of data samples into a SQL Server database. The first step is to create an IDL type definition. By using the **-example** option of *rtiddsgen* you will automatically create a Publishing application using this IDL type.

<NDDSHOME> is described in Paths Mentioned in Documentation (Section 1.1).

1. Create a new file named **MyType.idl** with the following contents:

```
struct MyType {
         short pkey; //@key
         char message[13];
};
```

   This IDL file specifies a data type that contains a message. Each instance is uniquely identified by the **pkey** field.

2. Assuming **<NDDSHOME>\bin** is in your **PATH** environment variable, execute the following command to compile the IDL type:

```
rtiddsgen -language C -example i86Win32VS2012 MyType.idl
```

This generates the **MyType**, **MyTypePlugin**, and **MyTypeSupport** files, as well as the **MyType_publisher** and **MyType_subscriber** example code.

3. The generated example will also have a Visual Studio Solution file called **MyType-vs2012.sln**. Start Microsoft Visual Studio and load this solution by opening this file.

4. Edit **MyType_publisher.c**, and find the line containing the comment:

```
/* Modify the data to be written here */
```

Insert the following lines immediately below this comment:

```
instance->pkey = count;
strcpy(instance->message, "Hello world!");
```

5. Save your changes and build the **MyType_publisher** project.

6. Start the **MyType_publisher** application so that it starts publishing data samples. From a command prompt with the working directory set to the location of the generated code, enter:

```
objs\i86Win32VS2005\MyType_publisher
```

On the screen, you will see:

```
Writing MyType, count 0
Writing MyType, count 1
Writing MyType, count 2
...
```

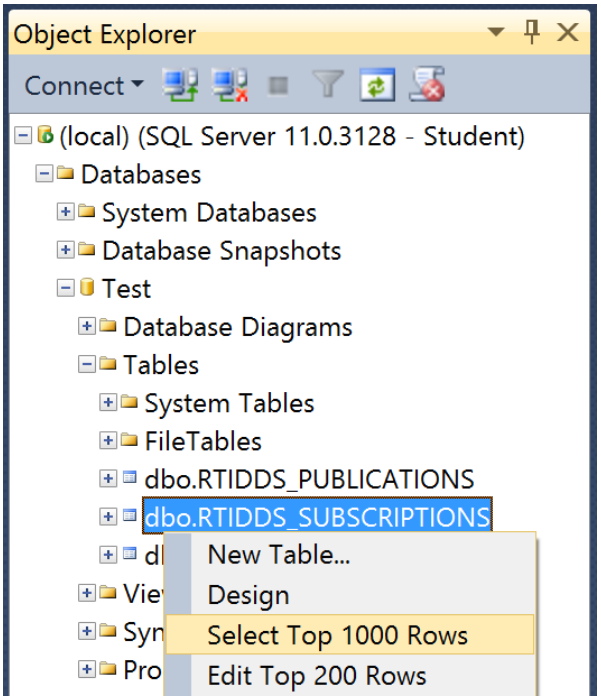
The samples are not captured in the SQL Server database yet. For this, you need to set up a subscription in *Database Integration Service*.

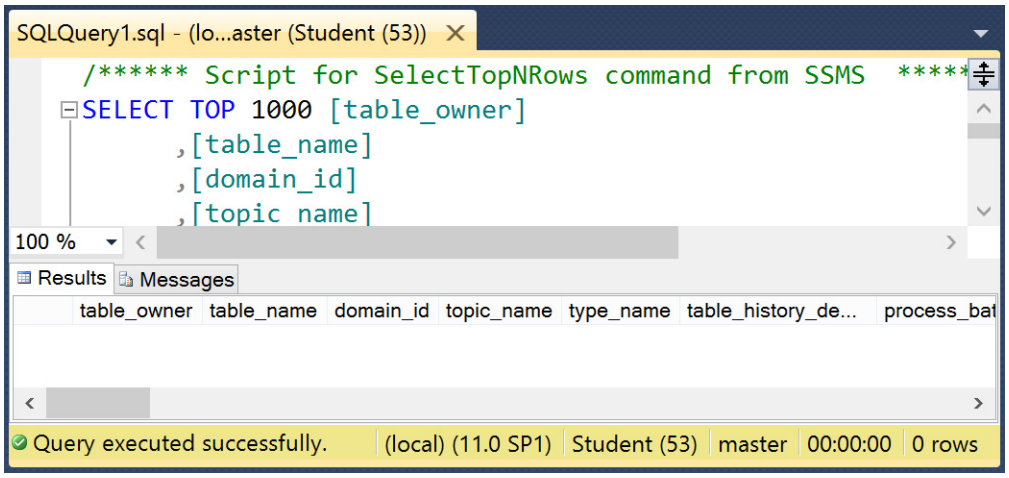
Subscriptions are set up in the **RTIDDS_SUBSCRIPTIONS** configuration table that *Database Integration Service* created when it connected to the database.

7. Start Microsoft SQL Server Management Studio and login as **Student**.

If you inspect the **RTIDDS_SUBSCRIPTIONS** configuration table, you will see that it is still empty at this point.

8. In the Object Explorer, expand **<*Your Server Name*>**, **Databases**, **Test**, and **Tables**. Right-click to open the context menu for **RTIDDS_SUBSCRIPTIONS** and choose **Select Top 1000 Rows**.





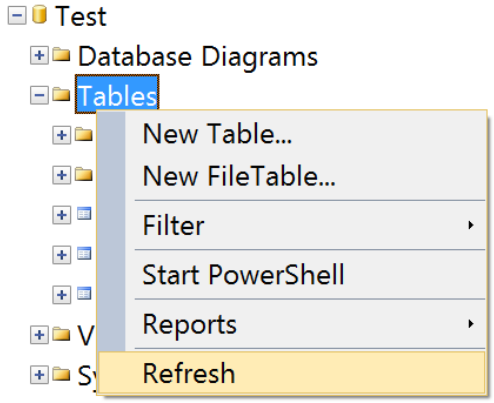
9. To store the samples from the **MyType_publisher** application in a table named **Example**, insert a corresponding entry into the **RTIDDS_SUBSCRIPTIONS** table. Right-click to open the context menu for **RTIDDS_SUBSCRIPTIONS** and choose **Edit Top 200 Rows**. In the first five columns, insert the values **Test**, **Example**, **0**, **Example MyType**, and **MyType**. Press **Enter**.

	table_owner	table_name	domain_id	topic_name	type_name	table_history_depth
	Test	Example	0	Example MyType	MyType	*NULL*
*	NULL	NULL	NULL	NULL	NULL	NULL

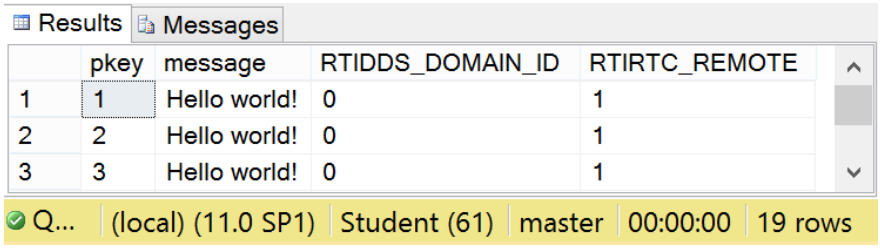
This entry directs *Database Integration Service* to create a user table named **Example** and to start storing samples published with topic **Example MyType** and data type **MyType** in Domain 0.

Note that the **table_owner** column requires a non-NULL value, but is not used for SQL Server.

10. If the **MyType_publisher** application is still running, you can use the Object Explorer as seen above to view the contents of the table—otherwise, restart **MyType_publisher** as described above before proceeding. You may need to refresh the Tables subtree in Object Explorer: right-click to open the context menu for Tables and select **Refresh**.



The output will look something like this:



The actual number of rows depends on when exactly the *Database Integration Service* Daemon started storing samples. If you execute the **SELECT** statement repeatedly you will see the number of rows grow. This is because the **MyType_publisher** application writes a new instance every four seconds.

9.9 Publishing Database Updates to Subscribers

In this section, you will enable *Database Integration Service* to automatically publish changes made to a table in a data source.

We assume that you have followed the instructions in Starting the Database Integration Service Daemon (Section 9.7). In addition, we assume that you have followed the instructions in Step 1 to Step 4 in Storing Samples from Publishers (Section 9.8) to create the IDL file, and generate and compile the example code.

1. If you have not already done so, stop the **MyType_publisher** application by pressing **Ctrl-C** in the window in which it is running.
2. Build the **MyType_subscriber** project in Visual Studio and start the application from the command line:

```
objs\i86Win32VS2012\MyType_subscriber
MyType subscriber sleeping for 4 sec...
MyType subscriber sleeping for 4 sec...
...
```

You will get this message repeatedly, since nothing is being published.

3. To publish changes from a database table, you need to configure the **RTIDDS_PUBLICATIONS** table. To publish the changes from the **Example** table, start Microsoft SQL Server Management Studio and login as **Student**. In the Object Explorer, expand ***<Your Server Name>***, **Databases**, **Test**, and **Tables**. Right-click to open the context menu for **RTIDDS_PUBLICATIONS** and choose **Edit Top 200 Rows**. In the first five columns, insert the values **Test**, **Example**, **0**, **Example MyType**, and **MyType**; press **Enter**.

   This entry directs *Database Integration Service* to start publishing changes to table **Example** as the topic **Example MyType** with type **MyType** in Domain 0.

4. Change one of the previously captured samples in the **Example** table. In the Object Explorer, right-click to open the context menu for the **Example** table and choose E**dit Top 200 Rows**. Scroll to the last row. Change the value in the message column to **Hello again!** Press **Enter**.

   You will see that the **MyType_subscriber** application reports the update, for example:

```
...
MyType subscriber sleeping for 4 sec...
pkey: 748
message:
   message[ 0]: 'H'
   message[ 1]: 'e'
   message[ 2]: 'l'
   message[ 3]: 'l'
   message[ 4]: 'o'
   message[ 5]: ' '
   message[ 6]: 'a'
   message[ 7]: 'g'
   message[ 8]: 'a'
   message[ 9]: 'i'
   message[10]: 'n'
   message[11]: '!'
   message[12]: <0>
MyType subscriber sleeping for 4 sec...
 ...
```

5. You can also update all entries in the table. In the Microsoft SQL Server Management Studio toolbar, click **New Query** and enter the following:

```
UPDATE [Example] SET message = 'Hello again!';
```

   In the toolbar, click **Execute Query**. Notice that the **MyType_subscriber** application receives all changes, possibly a large number.

Chapter 10 Database Integration Service for PostgreSQL on UNIX-Based Systems

10.1 Installing Database Integration Service

First, verify that PostgreSQL is installed and running on your system. The required version of PostgreSQL is listed in the *Release Notes*.

The installation of PostgreSQL is beyond the scope of this document. Please refer to the PostgreSQL installation manual to install and configure PostgreSQL.

Database Integration Service, requires the installation of the PostgreSQL ODBC driver. See the Release Notes for the required version.

The official ODBC driver and installation instructions are available at: https://odbc.postgresql.org/.

The PostgreSQL ODBC driver requires an ODBC driver manager. *Database Integration Service* requires UnixODBC, a complete, free/open ODBC solution for UNIX-based systems. You can download UnixODBC from http://www.unixodbc.org.

UnixODBC Note: *Database Integration Service* links to UnixODBC library **libodbc.so.1**. In release, 2.3.1, UnixODBC changed the library version from 1 to 2. If, after installing UnixODBC, *Database Integration Service* cannot find **libodbc.so**, create a symbolic link to **libodbc.so.1** from **libodbc.so.2**:

```
> ln -s libodbc.so.2 libodbc.so.1
```

To install Database Integration Service:

1. Install *Database Integration Service* on top of *RTI Connext DDS*. There are two ways to install it, from *RTI Launcher* or from the command line.

   <NDDSHOME> is described in Paths Mentioned in Documentation (Section 1.1).

   - To install from *RTI Launcher*:

     a. Start *RTI Launcher*:

     ```
     cd <NDDSHOME>/bin
     ./rtilauncher
     ```

     b. From the Utilities tab, click on **RTI Package Installer**.

     c. Use the + sign to add the *Database Integration Service* **.rtipkg** file that you want to install.

     d. Click **Install**.

   - To install from the command line:

     ```
     cd <NDDSHOME>/bin
     ```

```
./rtipkginstall <path to Database Integration Service .rtipkg file>
```

2. Add the path to the UnixODBC driver manager to the beginning of LD_LIBRARY_PATH. For example:

   ```
   > setenv LD_LIBRARY_PATH /usr/lib:$LD_LIBRARY_PATH
   ```

   or

   ```
   > export LD_LIBRARY_PATH=/usr/lib:$LD_LIBRARY_PATH
   ```

   Replace **/usr/lib** with the location of the UnixODBC driver manager on your system.

3. If your PostgreSQL ODBC driver is linked dynamically against PostgreSQL libraries, make sure that **libpq.so.5** is on your LD_LIBRARY_PATH before starting *Database Integration Service* for PostgreSQL.

10.2 Configuring PostgreSQL Server

1. Create a PostgreSQL account:

   Before you can use *Database Integration Service*, you need to get a PostgreSQL user account from your database administrator. If you are acting as your own database administrator, start **psql** (the PostgreSQL interactive terminal) from the command prompt to connect to the PostgreSQL server as the PostgreSQL root user.

   ```
   CREATE USER "MyUsername" WITH PASSWORD 'MyPassword';
   ```

   Note: PostgreSQL provides other mechanisms to create users and assign privileges (for example, the **createuser** command). The descriptions of these mechanisms are outside the scope of this document.

2. Create a database that is owned by the user/role created above.

   ```
   CREATE DATABASE "MyDatabase" WITH OWNER "MyUsername";
   ```

   Note: In this example we are making the user, MyUsername, the owner of the database so he can configure and manage it himself. If the user is not the database owner, the minimum set of privileges that are needed to work with *Database Integration Service* are permissions to connect to the database, to create tables, and to do SELECT, INSERT, and UPDATES on tables.

10.3 Create a Data Source for PostgreSQL

Database Integration Service uses the PostgreSQL ODBC driver through UnixODBC to access data sources. Usually these are the same data sources to which your applications connect. The connection information for each data source is stored in the **.odbc.ini** file. The stored information describes each data source in detail, specifying the driver name, a description, and any additional information the driver needs to connect to the data source.

To create the .odbc.ini file, follow these steps:

1. Create a new file named .odbc.ini in your home directory using your favorite text editor. Or you can use the ODBCINI environment variable to specify the file location.
2. Insert these lines in the file:

```
[MyPostgreSQLDsn]
Driver=/usr/lib/psqlodbca.so
UserName=MyUsername
Password=MyPassword
Database=MyDatabase
Servername=localhost
Port=5432
BoolsAsChar=0
```

   Notes:

   - Make sure that 'Driver' points to the valid location of the PostgreSQL ODBC driver on your system.
   - When connecting to a PostgreSQL server located on the local system, you can specify the port on which the server is listening with the Port attribute. If not present, the default value is 5432. Make sure that the Port value in the ODBC INI is valid.

3. Save your changes.

10.4 Creating a Configuration File for Database Integration Service

Database Integration Service reads its configuration information from a file. By default, *Database Integration Service* tries to load the configuration file, **<NDDSHOME>/resource/xml/RTI_REAL_TIME_CONNECT.xml**. You can specify a different file with the command-line option, -cfgFile.

The default file, **RTI_REAL_TIME_CONNECT.xml**, does not contain any actual valid information yet. For this example we will edit this file as follows:

1. Look for the tag <postgresql_connection>. Replace the tags <dsn>, <user_name>, and <password> as follows:

```
<postgresql_connection>
    <dsn>MyPostgreSQLDsn</dsn>
    <user_name>MyUsername</user_name>
    <password>MyPassword</password>
</postgresql_connection>
```

2. Save the file.

This configuration file instructs *Database Integration Service* to monitor the data source as specified by the "MyPostgreSQLDsn" DSN.

10.4.1 Starting the Database Integration Service Daemon

Start *Database Integration Service* by executing the following commands.

<NDDSHOME> is described in Paths Mentioned in Documentation (Section 1.1).

```
> cd <NDDSHOME>/bin
> ./rtirtc_postgresql -noDaemon -cfgName default
```

By default, *Database Integration Service* runs in the background as a daemon process. However, using the **-noDaemon** option prevents that and starts the *Database Integration Service* Daemon as a regular process. Messages are sent to standard output.

You should see the following output, indicating that the process is running.

```
>./rtirtc_postgresql -noDaemon -cfgName default
RTI Database Integration Service to PostgreSQL, Release 5.x.y.z: startup
succeeded
Database Integration Service is now connected to the "MyPostgreSQLDsn" data
source.
```

Notes:

- Make sure you have the UnixODBC driver manager library, **libodbc.so**, in your LD_LIBRARY_-PATH.
- You can optionally use the command-line option **-verbosity 3** for debugging purposes and to get information on what the *Database Integration Service* is doing.

10.4.2 Storing Samples from Publishers

Database Integration Service for PostgreSQL does not currently support creating new subscriptions after startup. Make sure you stop the *Database Integration Service* daemon by pressing CTRL-C before continuing.

1. Create a new text file called **MyType.idl** with the following contents:

   ```
   struct MyType {
       short pkey; //@key
       string message;
   };
   ```

   This IDL file specifies a data type that contains a message. Each instance is uniquely identified by the **pkey** field.

2. Execute the following command to compile the IDL type.

   <NDDSHOME> is described in Paths Mentioned in Documentation (Section 1.1).

   ```
   > <NDDSHOME>/bin/rtiddsgen -language C -example <arch> MyType.idl
   ```

   For example:

   ```
   > <NDDSHOME>/bin/rtiddsgen -language C -example i86Linux2.6gcc4.1.1
     MyType.idl
   ```

   This generates the MyType, MyTypePlugin, and MyTypeSupport files, as well as the MyType_-publisher and MyType_subscriber example code.

3. The generated example will also have a makefile named **makefile_MyType_<arch>**.

   You may need to edit the makefile to specify the location of the compiler if it is not available on your path.

4. Edit **MyType_publisher.c,** and find the line containing the comment:

   ```
   /* Modify the data to be written here */
   ```

   Insert the following lines immediately below this comment:

   ```
   instance->pkey = count;
    strcpy(instance->message, "Hello world!");
   ```

5. Save your changes and build the MyType_publisher and MyType_subscriber applications by executing:

   ```
   > gmake -f makefile_MyType_<arch>
   ```

6. Start the MyType_publisher application so that it starts publishing data samples.

   ```
   > objs/<arch>/MyType_publisher
   ```

   On the screen, you will see:

   ```
    Writing MyType, count 0
    Writing MyType, count 1
    Writing MyType, count 2
    ...
   ```

   The samples are not captured in the PostgreSQL database yet. For this, you need to set up a subscription in *Database Integration Service*.

   Subscriptions are set up in the "RTIDDS_SUBSCRIPTIONS" configuration table that *Database Integration Service* created when it connected to the PostgreSQL database for the first time. Start postgreSQL client from the command prompt:

   ```
   ><PostgreSQL installation dir>/bin/psql MyDatabase -U MyUsername
   Password for user MyUsername:
   psql (9.5.2)
   Type "help" for help.

    MyDatabase=>
   ```

   Note: Make sure you start the client using the same username and database specified in the *Database Integration Service* XML configuration file.

7. You can see that the table RTIDDS_SUSCRIPTIONS is still empty at this point. You can verify it by running this command at the postgreSQL client prompt:

   ```
   MyDatabase=> select * from rtidds_subscriptions;
   ```

8. To store the samples from the MyType_publisher application in a table named "Example", insert a corresponding entry into the "RTIDDS_SUBSCRIPTIONS" table:

   ```
   MyDatabase=> INSERT INTO rtidds_subscriptions (table_owner, table_name,
    domain_id, topic_name, type_name, table_schema) VALUES
    ('MyDatabase','Example',0,'Example MyType','MyType','JSONB'); INSERT 0 1
   ```

   This command will create a subscription to the topic **Example MyType** of type **MyType** on domain 0. The received samples will be stored in the table **Example** in JSON format using a JSONB column.

   Notes:

   - **table_owner** in PostgreSQL has the format **<database_name>[.<schema_name>]**. If the schema is not provided, *Database Integration Service* assumes the default, **public**.
   - **table_schema** is used to configure the format in which the samples will be stored in the database. PostgreSQL supports three formats: FLATTEN, JSON, and JSONB. For additional information, see the *User's Manual*.

   Alternatively, you can create the subscription using the *Database Integration Service* XML configuration file. To do this, add a subscription entry under the subscription tags in the **postgresql_connection** section in the XML file.

   ```
   <postgresql_connection>
       <dsn>MyPostgreSQLDsn</dsn>
       <user_name>MyUsername</user_name>
       <password>MyPassword</password>
       <subscriptions>
           <subscription>
               <table_owner>MyDatabase</table_owner>
               <table_name>Example</table_name>
               <domain_id>0</domain_id>
               <topic_name>Example MyType</topic_name>
   ```

```
            <type_name>MyType</type_name>
            <table_schema>JSONB</table_schema>
        </subscription>
    </subscriptions>
</postgresql_connection>
```

9. Start *Database Integration Service* as described in previous section.

```
> cd <NDDSHOME>/bin
 > ./rtirtc_postgresql -noDaemon -cfgName default
```

10. If the MyType_publisher application is still running, you can execute the following SQL statement to view the contents of the table—otherwise, restart MyType_publisher as described above before executing this statement.

```
MyDatabase=> select * from example;
```

```
rtidds_keyhash                    | payload                              | rtidds_domain_id |rtirtc_remote
----------------------------------+--------------------------------------+------------------+-------------
\x0001000000000000000000000000000 | {"pkey": 1, "message": "Hello World!"} |       129        |     1
\x0002000000000000000000000000000 | {"pkey": 2, "message": "Hello World!"} |       129        |     1
\x0003000000000000000000000000000 | {"pkey": 3, "message": "Hello World!"} |       129        |     1
(3 rows)
```

The actual number of rows found depends on exactly when the *Database Integration Service* Daemon started storing samples. If you execute the “select” statement repeatedly, you will see the number of rows grow. This is because the MyType_publisher application writes a new instance every 4 seconds.