RTI Database Integration Service

Getting Started Guide

Version 6.1.0

Printed in U.S.A. First printing.
April 2021.

Trademarks

Copy and Use Restrictions

Technical Support

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

Contents

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

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* requires that *Connext DDS* already be installed.

To use MySQL, read either:

- Chapter 2 Database Integration Service for MySQL on Linux Systems on page 4
- Chapter 3 Database Integration Service for MySQL on Windows Systems on page 14

To use PostgreSQL®, read:

- Chapter 4 Database Integration Service for PostgreSQL on Linux Systems on page 24

1.1 Paths Mentioned in Documentation

The documentation refers to:

- **<NDDSHOME>**

  This refers to the installation directory for *RTI® Connext® DDS*. The default installation paths are:
  - macOS® systems:
    /Applications/rti_connext_dds-6.1.0
  - Linux systems, non-*root* user:
    /home/<*your user name*>/rti_connext_dds-6.1.0
  - Linux systems, *root* user:
    /opt/rti_connext_dds-6.1.0
  - Windows® systems, user without Administrator privileges:
    <*your home directory*>\rti_connext_dds-6.1.0
  - Windows systems, user with Administrator privileges:
    C:\Program Files\rti_connext_dds-6.1.0

  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-6.1.0\bin\rtiddsgen"
  ```

  Or if you have defined the **NDDSHOME** environment variable:

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

- ***<path to examples>***

By default, 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 the copied examples as ***<path to examples>***.

Wherever you see ***<path to examples>***, replace it with the appropriate path.

Default path to the examples:

- macOS systems: **/Users/<*your user name*>/rti_workspace/6.1.0/examples**
- Linux systems: **/home/<*your user name*>/rti_workspace/6.1.0/examples**
- Windows systems: **<*your Windows documents folder*>\rti_workspace\6.1.0\examples**

  Where 'your Windows documents folder' depends on your version of Windows. For example, on Windows 10, the folder is **C:\Users\<*your user name*>\Documents**.

Note: You can specify a different location for **rti_workspace**. You can also specify that you do not want the examples copied to the workspace. For details, see *Controlling Location for RTI Workspace and Copying of Examples* in the *RTI Connext DDS Installation Guide*.

1.2 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, and supported architectures. (For migration and compatibility information, see the Migration Guide on the RTI Community Portal: https://community.rti.com/documentation.)
- *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*.

1.2.1 Additional Resources

- The *ODBC API Reference* from Microsoft is available from http://msdn.microsoft.com/en-us/library/ms714562(VS.85).aspx.
- Documentation for MySQL: http://dev.mysql.com/doc/#manual.
- Documentation for PostgreSQL: https://www.postgresql.org/docs.

Chapter 2 Database Integration Service for MySQL on Linux Systems

2.1 Before Installing

Before you can run *Database Integration Service*:

- Verify that MySQL is installed and running on your system. See Supported Platforms and System Requirements, in the RTI Database Integration Service Release Notes for version information. 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 ODBC Driver Requirements, in the RTI Database Integration Service Release Notes for version information. 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/.

  The installation guide can be found in https://dev.mysql.com/doc/connector-odb-c/en/connector-odbc-installation.html.
- The MySQL ODBC driver requires an ODBC driver manager. We recommend using UnixODBC, a complete, free/open ODBC solution for Linux systems. See ODBC Driver Requirements, in the RTI Database Integration Service Release Notes. The driver manager can be downloaded from http://www.unixodbc.org.
- **Note for UnixODBC:** *Database Integration Service* links to the UnixODBC library **libod-bc.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 sym-

link to **libodbc.so.1** from **libodbc.so.2**:

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

2.2 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 .

   To install from *RTI Launcher*:

   a. Start *RTI Launcher*:

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

   b. From the **Configuration** tab, click on **Install RTI Packages**.

   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.

2.3 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 **resource/app/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**.

2.3.1 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/5.7/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/5.7/en/install-plugin.html.

2.3.2 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 *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 the RTI Connext DDS Installation Guide.

2.4 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 '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.

2.5 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.

See ODBC Driver Requirements, in the RTI Database Integration Service Release Notes for supported versions of MySQL ODBC.

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/libmyodbc5.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.

2.6 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.

2.7 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.

2.8 Starting the Database Integration Service Daemon

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

<NDDSHOME> is described in 1.1 Paths Mentioned in Documentation on page 2.

```
> 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 6.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 2.2 Installation on page 5 and see 2.1 Before Installing on page 4).

2.9 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]Please see the *RTI Code Generator User's Manual* for more information about how to run *rtiddsgen*.

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.

   <NDDSHOME> is described in 1.1 Paths Mentioned in Documentation on page 2.

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

   For example:

```
> <NDDSHOME>/bin/rtiddsgen -language C -example x64Linux2.6gcc4.4.5 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, 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.

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.

2.10 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 2.8 Starting the Database Integration Service Daemon on page 9 to start a *Database Integration Service* Daemon. In addition, we assume that you have followed the instructions in Steps 1 to 4 in 2.9 Storing Samples from Publishers on page 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, 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, 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 3 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 2017[1].

3.1 Before Installing

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

- The MySQL ODBC driver; see ODBC Driver Requirements, in the RTI Database Integration Service Release Notes for version information. 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/

  The installation guide can be found here:
  https://dev.mysql.com/doc/connector-odbc/en/connector-odbc-installation.html
- Visual C++ Redistributable for Visual Studio 2012.

  It is available from this Microsoft website:

  https://www.microsoft.com/en-us/download/details.aspx?id=30679
- MySQL (must be installed and running).

[1] You may use other supported Microsoft compilers such as Visual Studio 2012 or Visual Studio 2015, however, the instructions in this chapter were written assuming Visual Studio 2017.

See Supported Platforms and System Requirements, in the RTI Database Integration Service 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.

3.2 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 1.1 Paths Mentioned in Documentation on page 2.

From *RTI Launcher*:

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

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

2. From the **Configuration** tab, click on **Install RTI Packages**.
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>
```

3.3 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 **resource\app\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**.

3.3.1 Installing rtirtc_mysqlq.dll

Copy **<NDDSHOME>resource\app\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 5.7\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/5.7/en/install-plugin.html.

3.3.2 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>\resource\app\lib\<*arch*>**).

If **<NDDSHOME>\resource\app\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 *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 the RTI Connext DDS Installation Guide.

3.4 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.

[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).

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.

3.5 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.

See ODBC Driver Requirements, in the RTI Database Integration Service Release Notes for supported versions of MySQL ODBC.

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).** (In some Windows versions, the application name might be different: for example, "ODBC Data Sources.")
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*.
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.

3.6 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.

3.7 Using Database Integration Service as a Windows Service

3.7.1 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
```

3.7.2 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 start rtirtc_mysql600
```

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 stop rtirtc_mysql600
```

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**.

3.7.3 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
```

3.8 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.

3.9 Starting the Database Integration Service Daemon

You can start the *Database Integration Service* Daemon as a Windows service (assuming you followed the steps in 3.7 Using Database Integration Service as a Windows Service on the previous page). 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.

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

```
RTI Database Integration Service for MySQL, Release 6.x.y: startup succeeded
```

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-5.7]
Can't open shared library 'rtirtc_mysqlq.dll' (errno: 2)
```

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

```
[DDSQLDaemonCNAHelper_createCNAConnection,line
761:ERROR:4096:50002] [CNA:CNAOpe n] Error initializing MySQL Server
Queue: [MySQL][ODBC 5.1.6 Driver][mysqld-5.7]
FUNCTION test.MySqlNddsQueue_initialize does not exist
```

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

3.10 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 x64Win64VS2017[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 value here, such as x64Win32VS2015 for Visual Studio 2015.

3. The generated example will also have a Visual Studio Solution file called **MyType-vs2017.sln**. Start Microsoft Visual Studio 2017 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_s(instance->message, 13, "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, enter:

```
> objs\x64Win64VS2017\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> 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.

3.11 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 3.9 Starting the Database Integration Service Daemon on page 19 to start a *Database Integration Service* Daemon. In addition, we assume that you have followed the instructions in Steps 1 to 4 in 3.10 Storing Samples from Publishers on page 20 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\x64Win64VS2017\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 4 Database Integration Service for PostgreSQL on Linux Systems

4.1 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 Linux 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 1.1 Paths Mentioned in Documentation on page 2.

To install from *RTI Launcher*:

a. Start *RTI Launcher*:

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

b. From the **Configuration** tab, click on **Install RTI Packages**.

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.

4.2 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.

4.3 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.

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>/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.

4.4.1 Starting the Database Integration Service Daemon

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

<NDDSHOME> is described in 1.1 Paths Mentioned in Documentation on page 2.

```
> 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 6.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.

4.4.2 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 .

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

   For example:

```
> <NDDSHOME>/bin/rtiddsgen -language C -example x64Linux2.6gcc4.4.5 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
-----------------------------------+----------------------------------------+------------------+-------------
\x0001000000000000000000000000000000 | {"pkey": 1, "message": "Hello World!"} |       129        |     1
\x0002000000000000000000000000000000 | {"pkey": 2, "message": "Hello World!"} |       129        |     1
\x0003000000000000000000000000000000 | {"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.