RTI Queuing Service

Getting Started Guide

Version 5.2.0

Your systems. Working as one.

Printed in U.S.A. First printing.
June 2015.

Technical Support

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

Contents

Chapter 1 Welcome to RTI Queuing Service

RTI® Queuing Service is a broker that provides a queuing communication model in which a sample is stored in a queue until it is consumed by one QueueConsumer. If there are no QueueConsumers available when the sample is sent, the sample is kept in the queue until a QueueConsumer is available to process it. If a QueueConsumer receives a sample and does not acknowledge it before a specified amount of time or acknowledges it negatively, the sample will be redelivered to a different QueueConsumer.

Queuing Service provides an "at-most-once" and "at-least once" delivery semantic.

By default, *Queuing Service* keeps the samples in memory. To provide fault tolerance, *Queuing Service* can be configured to keep the samples on disk.

For high availability, *Queuing Service* provides mechanisms to replicate its state so that samples can survive the loss of any particular service and/or computer.

Chapter 2 Installing Queuing Service

This chapter describes:

- ❏ Paths Mentioned in Documentation (Section 2.1)
- ❏ Installing on a UNIX-Based System (Section 2.2)
- ❏ Installing on a Windows System (Section 2.3)

2.1 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.

2.2 Installing on a UNIX-Based System

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

From RTI Launcher:

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

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

   <NDDSHOME> is described in Paths Mentioned in Documentation (Section 2.1).

2. From the Utilities tab, select **RTI Package Installer**.
3. In the resulting dialog, 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>
```

If you want to install *Queuing Service* without user interaction (unattended mode), use the **-u** flag when installing from the command line:

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

Queuing Service will be installed in the <NDDSHOME> directory (see Paths Mentioned in Documentation (Section 2.1)).

2.3 Installing on a Windows System

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

From RTI Launcher:

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

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

   <NDDSHOME> is described in Paths Mentioned in Documentation (Section 2.1).

2. From the Services tab, select **Queuing Service**.

3. In the resulting dialog, 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>
```

If you want to install *Queuing Service* without user interaction (unattended mode), use the **-u** flag when installing from the command line:

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

Queuing Service will be installed in the <NDDSHOME> directory (see Paths Mentioned in Documentation (Section 2.1)).

Chapter 3 Using the Examples

Queuing Service includes two examples to show its most relevant functionality:

- **hello_world**: A Hello World application, in which is shown how to send/receive samples from/to *Queuing Service*. The example also shows how to use other relevant features such as persistence and replication.
- **remote_config**: A Remote Configuration example, in which is shown how to remotely create/delete resources, query their status, get a message, or flushing SharedReaderQueues. This example uses the Request/Reply API.

The examples are in **<path to examples>/queuing_service/<language>**, where <path to examples> is described in Paths Mentioned in Documentation (Section 2.1) and <language> is **c++** for C++ or **cs** for .NET. There are some differences between the versions:

- The .NET **hello_world** example uses the *Queuing Service* wrapper API, while the C++ example uses *DataWriters* and *DataReaders* directly to interact with *Queuing Service*, since the wrapper API is not available for C++.
- The .NET **hello_world** example uses two SharedReaderQueues: a request and a reply SharedReaderQueue. The C++ example only uses a request SharedReaderQueue.
- The .NET **hello_world** example is also a performance test, measuring requests and replies per second, The C++ version sends one message per second.

By default, the .NET **hello_world** example's SharedReaderQueues use different types than the C++ example.

Because of these differences, you will need to make some modifications in the examples in order for a **hello_world** C++ Producer to interoperate with a **hello_world** .NET Replier, and vice-versa.

To run the examples, please follow the instructions in the **README.txt** file included in the example's directory.

Chapter 4 Running Queuing Service

Queuing Service runs as a separate application. The script to run the executable is in **<NDDSHOME>/bin**. There are four ways to start *Queuing Service*:

- ❏ Starting from Launcher (Section 4.1)
- ❏ Starting Manually from the Command Line (Section 4.2)
- ❏ Using Queuing Service as a Windows Service (Section 4.3)

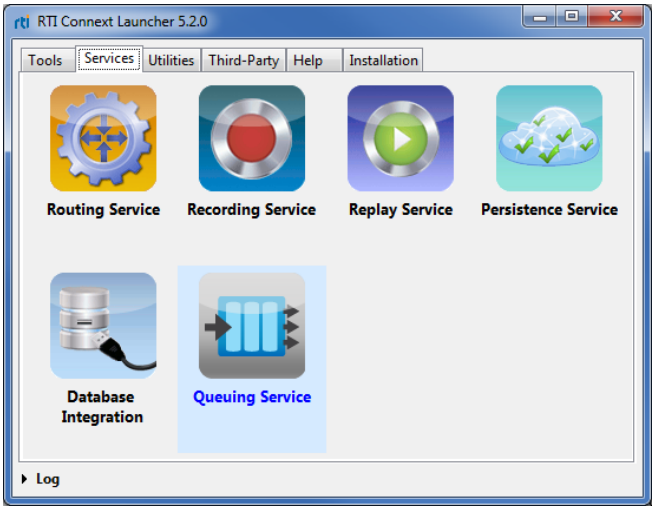
If you are starting *Queuing Service* as a Windows Service, also read Notes when Running as a Windows Service (Section 4.3.3).

4.1 Starting from Launcher

1. Start *RTI Launcher* from the Start menu (on Windows systems) or on the command line, type:

   ```
   <NDDSHOME>/bin/rtilauncher
   ```

2. From the Services tab, select **Queuing Service:**



4.2 Starting Manually from the Command Line

To start Queuing Service, enter:

```
cd <NDDSHOME>
bin/rtiqueuingservice [options]
```

Example:

```
cd <NDDSHOME>
bin/rtiqueuingservice -cfgFile example.xml -cfgName QueuingService_1
```

Table 4.1 describes the command-line options.

Table 4.1 **RTI Queuing Service Command-Line Options**

Option	Description
-appName <*name*>	Assigns a name to the execution of *Queuing Service*. Remote commands will refer to the queuing service using this name. In addition, the name of *DomainParticipants* created by *Queuing Service* will be based on this name. Default: The name given with **-cfgName**, if present, otherwise it is **RTI_Queuing_Service**.
-cfgFile <*name*>	Specifies a configuration file to be loaded. This parameter is required. See Section 3.1, How to Load the XML Configuration, in the *Queuing Service User's Manual*.
-cfgName <*name*>	Specifies a configuration name. *Queuing Service* will look for a matching **<queuing_service>** tag in the configuration file. **This parameter is required unless -cfgRemote is used.**
-cfgRemote	Specifies that the initial configuration of the service must be obtained remotely from other running instances. Using this option also requires the use of **-remoteAdministrationDomainId** to enable remote administration, because the initial configuration will be received in the remote administration domain ID. If you use this option and **-cfgName**, the service will wait until a configuration with that name is received. Otherwise, the service will use the first configuration that it receives. If the service does not receive the initial configuration after a configurable timeout (see **-cfgRemoteTimeout**), it will load the configuration from the input configuration file(s).
-cfgRemoteTimeout <*n*>	Specifies the maximum amount of time, in seconds, that *Queuing Service* will wait for an initial configuration when using **-cfgRemote**. Default: 20 seconds
-daemon	Runs *Queuing Service* as a daemon/Windows service. When this flag is present, *Queuing Service* will start in the background. Note that some systems may require special privileges to do this.
-domainIdBase <*ID*>	Sets the base domain ID. This value is added to the domain IDs in the configuration file. For example, if you set **–domainIdBase** to 50 and use domainIDs 0 and 1 in the configuration file, then *Queuing Service* will use domains 50 and 51. Default: 0
-help	Displays help information.

Table 4.1 **RTI Queuing Service Command-Line Options**

Option	Description
-remoteAdministrationDomainId <*ID*>	Enables remote administration and sets the domain ID for remote communication. When remote administration is enabled, *Queuing Service* will create a *DomainParticipant*, *Publisher*, *Subscriber*, *DataWriter*, and *DataReader* in the designated domain. See Chapter 5, Administering *Queuing Service* from a Remote Location, in the *Queuing Service User's Manual*. This option overrides the value of the tag **<domain_id>** within a **<administration>** tag. This parameter is required when using **-cfgRemote**. Default: Remote administration is not enabled unless it is enabled from the XML file.
-persistentFilePrefix	Specifies a name prefix to use with all files created by *Queuing Service*. This option overrides the value of the tag **<file_prefix>** within **<persistence_settings>/<filesystem>**. Default: Value in **<persistence_settings>/<filesystem>/<file_prefix>**.
-persistentStoragePath	Configures the directory for persistent storage. This option overrides the value of the tag **<directory>** within **<persistence_settings>/<filesystem>**. Default: Value in **<persistence_settings>/<filesystem>/<directory>**.
-var <*name*>=<*value*>	Sets the value of the variable <*name*>. This variable can be referenced within the XML configuration files using the **$(<name>)** notation. See Section 3.4, Using Variables in XML, in the *Queuing Service User's Manual* for more information on configuration variables. You may have more than one **-var** flag on the command line. On Windows platforms, you will need to put quotation marks around the variable name and value, like this: `-var "MY_VAR=myvalue"`
-verbosity <*n*>	Controls what type of messages are logged: 0 - Silent 1 - Exceptions (*Connext DDS* and *Queuing Service*) (default) 2 - Warnings (*Queuing Service*) 3 - Information (*Queuing Service*) 4 - Warnings (*Connext DDS* and *Queuing Service*) 5 - Tracing (*Queuing Service*) 6 - Tracing (*Connext DDS* and *Queuing Service*) Each verbosity level, *n*, includes all the verbosity levels smaller than *n*.
-version	Prints the *Queuing Service* version number.

4.3 Using Queuing Service as a Windows Service

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

4.3.1 Enabling Queuing Service to Run as a Windows Service

If you want to run *Queuing Service* 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\rtiqueuingservice -installService
```

By default, *Queuing Service* is installed with the service name **rtiqueuingservice520**. If you want to install it with a different service name, you can use the **-serviceName** flag. For instance (you would enter this all on one line):

```
<NDDSHOME>\bin\rtiqueuingservice -installService
   -serviceName mycustomservicename
```

Using the **-serviceName** parameter with different names allows you to install multiple *Queuing Service* instances on the same host.

4.3.2 Running RTI Queuing Service as a Windows Service

If you added *Queuing Service* as a Windows Service and want to run it without rebooting, you can start it as a service from the command line with the Windows **sc** utility:

```
sc <serviceName> start
```

By default, it will start *Queuing Service* with the "defaultService" configuration that is stored in **<NDDSHOME>\resource\xml\RTI_QUEUING_SERVICE.xml**. This configuration contains a service running with an empty SharedSubscriber with remote administration and monitoring enabled.

If you want to start *Queuing Service* with different parameters, you can use the utility **nssm**. You can specify the parameters from the command line by setting the option **AppParameters**. For example (you would enter this all on one line):

```
%NDDSHOME%\resource\app\bin\x64Win64VS2008\nssm.exe set <serviceName>
AppParameters "<queuing service arguments>"
```

For more information and examples, see Notes when Running as a Windows Service (Section 4.3.3).

Additionally, you can start *Queuing Service* 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**.

4.3.3 Notes when Running as a Windows Service

Here are some things to consider when running *Queuing Service* as a Windows Service:

- ❏ All AppParameters arguments must be enclosed in quotation marks.
- ❏ To refer to variables in the XML configuration file, use the *Queuing Service* command-line option **-var** to set the variable's value. The syntax for referring to a variable in the XML file is:

  ```
  <name>$(NAME)</name>
  ```

- For the AppParameters passed to nssm, use **-var** like this:

  ```
  -var MY_DOMAIN=10
  ```

  For example (you would enter this all on one line):

  ```
  %NDDSHOME%\resource\app\bin\x64Win64VS2008\nssm.exe set
  rtiqueuingservice520 AppParameters
  "-cfgFile \"C:\dir with spaces\qsconf-with-vars.xml\"
   -cfgName MyCustomConf -var MY_DOMAIN=10"
  ```

- If a variable value includes spaces, you must enclose the value in escaped quotes. For example (you would enter this all on one line):

  ```
  %NDDSHOME%\resource\app\bin\x64Win64VS2008\nssm.exe
  set rtiqueuingservice520 AppParameters
  "-cfgFile \"C:\dir with spaces\qsconf-with-vars.xml\"
   -cfgName MyCustomConf -var \"NAME=My QS name\""
  ```

- If you use environment variables instead of the **-var** command-line option, you may need to restart your Windows machine.

- If you specify **-cfgFile** in the Start Parameters field, you must use the full path to the file.

- Some versions of Windows do not allow Windows Services to communicate with other services/applications using shared memory. For this reason, if you plan to run *Queuing Service* as Windows Service, you should disable the shared-memory transport in all the DomainParticipants created by *Queuing Service* and in the applications communicating with *Queuing Service*. For more information on setting builtin transports, see the *RTI Connext DDS Core Libraries User's Manual* (Section 15.1, Builtin Transport Plugins).

- In some scenarios, you may need to add a multicast address (e.g., builtin.udpv4://239.255.0.1) to your discovery peers. For details on setting the discovery peers, see the *RTI Connext DDS Core Libraries Getting Started Guide* (Section 4.1.2, How to Set Your Discovery Peers).

4.3.4 Stopping Queuing Service when it is Running as a Windows Service

To stop *Queuing Service* when it is running as a Windows Service, use this command:

```
sc rtiqueuingservice520 stop
```

You can also start/stop *Queuing Service* 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**.

4.3.5 Disabling Queuing Service from Running as a Windows Service

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

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

By default, the service **rtiqueuingservice520** is uninstalled. If you want to uninstall a different service instance, add the **-serviceName** option to the above command.