# RTI Queuing Service

Getting Started Guide

Version 6.1.0

**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 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.

## 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*.

# Chapter 2 Installing Queuing Service

This chapter describes:

- 2.1 Installing on a Linux or macOS System below
- 2.2 Installing on a Windows System on the next page

## 2.1 Installing on a Linux or macOS 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 1.1 Paths Mentioned in Documentation on page 1.
2. From the **Configuration** tab, select **Install RTI Packages**.
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 1.1 Paths Mentioned in Documentation on page 1).

## 2.2 Installing on a Windows 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 Start menu or the command line:

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

   <NDDSHOME> is described in 1.1 Paths Mentioned in Documentation on page 1.
2. From the **Configuration** tab, select **Install RTI Packages**.
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 1.1 Paths Mentioned in Documentation on page 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 1.1 Paths Mentioned in Documentation on page 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 three ways to start *Queuing Service*:

- 4.1 Starting from Launcher below
- 4.2 Starting Manually from the Command Line below
- 4.3 Using Queuing Service as a Windows Service on page 9

If you are starting *Queuing Service* as a Windows Service, also read 4.3 Using Queuing Service as a Windows Service on page 9.

## 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
```

To run this service executable on a *target* system (not your host development platform), you must first select the target architecture. To do so, either:
Set the environment variable **CONNEXTDDS_ARCH** to the name of the target architecture. (Do this for each command shell you will be using.)
Or set the variable **connextdds_architecture** in the file **rticommon_config.[sh/bat]**[a] to the name of the target architecture. If the **CONNEXTDDS_ARCH** environment variable is set, the architecture in this file will be ignored.

Table 4.1 RTI Queuing Service Command-Line Options 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*.<br>Remote commands will refer to the queuing service using this name.<br>In addition, the name of *DomainParticipants* created by *Queuing Service* will be based on this name.<br>Default: The name given with **-cfgName**, if present, otherwise it is **RTI_Queuing_Service**. |
| -cfgFile <*name*> | Specifies a configuration file to be loaded.<br>This parameter is required.<br>See Section 3.1 How to Load the XML Configuration from a File 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.<br>**This parameter is required unless -cfgRemote is used.** |
| -cfgRemote | Specifies that the initial configuration of the service must be obtained remotely from other running in-stances.<br>Using this option also requires the use of **-remoteAdministrationDomainId** to enable remote ad-ministration, because the initial configuration will be received in the remote administration domain ID.<br>If you use this option and **-cfgName**, the service will wait until a configuration with that name is re-ceived. Otherwise, the service will use the first configuration that it receives.<br>If the service does not receive the initial configuration after a configurable timeout (see **-cfgRe-moteTimeout**), 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 con-figuration when using **-cfgRemote**.<br>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. |

[a]This file is resource/scripts/rticommon_config.sh on Linux or macOS systems, resource/scripts/rticommon_config.bat on Windows systems.

## Table 4.1 RTI Queuing Service Command-Line Options

| Option | Description |
|---|---|
| -domainIdBase <*ID*> | Sets the base domain ID.<br>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.<br>Default: 0 |
| -heapSnapshotPeriod | Enables heap monitoring.<br>*Queuing Service* will generate a heap snapshot every <sec>.<br>Default: heap monitoring is disabled. |
| -heapSnapshotDir | When heap monitoring is enabled, this parameter configures the directory where the snapshots will be stored. The snapshot filename format is RTI_<configurationName><processId><index>.log.<br>Default: current working directory |
| -help | Displays help information. |
| -remoteAdministrationDomainId <*ID*> | Enables remote administration and sets the domain ID for remote communication.<br>When remote administration is enabled, *Queuing Service* will create a *DomainParticipant*, *Publisher*, *Subscriber*, *DataWriter*, and *DataReader* in the designated domain.<br>See Chapter 5, Administering Queuing Service from a Remote Location, in the *Queuing Service User's Manual*.<br>This option overrides the value of the tag **<domain_id>** within a **<administration>** tag.<br>This parameter is required when using **-cfgRemote**.<br>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*.<br>This option overrides the value of the tag **<file_prefix>** within **<persistence_ settings>/<filesystem>**.<br>Default: Value in **<persistence_settings>/<filesystem>/<file_prefix>**. |
| -persistentStoragePath | Configures the directory for persistent storage.<br>This option overrides the value of the tag **<directory>** within **<persistence_settings>/<filesystem>**.<br>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.<br>You may have more than one **-var** flag on the command line.<br>On Windows platforms, you will need to put quotation marks around the variable name and value, like this:<br>-var "MY_VAR=myvalue" |

Table 4.1 RTI Queuing Service Command-Line Options

| Option | Description |
|---|---|
| -verbosity <*n*> | Controls what type of messages are logged:<br>0 - Silent<br>1 - Exceptions (*Connext DDS* and *Queuing Service*) (default)<br>2 - Warnings (*Queuing Service*)<br>3 - Information (*Queuing Service*)<br>4 - Warnings (*Connext DDS* and *Queuing Service*)<br>5 - Tracing (*Queuing Service*)<br>6 - Tracing (*Connext DDS* and *Queuing Service*)<br>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. If you want to run *Queuing Service* as a Windows Service, use a Windows service wrapper such as **nssm** or **winsw**. For instance, you can download **nssm** from https://nssm.cc/download. Follow the product's documentation to set up *Queuing Service* as a Windows service. For example, for **nssm**, see https://nssm.cc/usage.

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

- 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 a 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 Builtin Transport Plugins, in the RTI Connext DDS Core Libraries User's Manual.
- 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 information about setting discovery peers in the "Troubleshooting" section of *Introduction to Publish/Subscribe*, in the RTI Connext DDS Getting Started Guide.