

RTI Queuing Service

Getting Started Guide

Version 7.3.0



Trademarks

RTI, Real-Time Innovations, Connex, Connex Drive, NDDS, the RTI logo, 1RTI and the phrase, “Your Systems. Working as one.” are registered trademarks, trademarks or service marks of Real-Time Innovations, Inc. All other trademarks belong to their respective owners.

Copy and Use Restrictions

No part of this publication may be reproduced, stored in a retrieval system, or transmitted in any form (including electronic, mechanical, photocopy, and facsimile) without the prior written permission of Real-Time Innovations, Inc. The software described in this document is furnished solely under and subject to RTI's standard terms and conditions available at <https://www.rti.com/terms> and in accordance with your License Acknowledgement Certificate (LAC) and Maintenance and Support Certificate (MSC), except to the extent otherwise accepted in writing by a corporate officer of RTI.

Third-Party Software

RTI software may contain independent, third-party software or code that are subject to third-party license terms and conditions, including open source license terms and conditions. Copies of applicable third-party licenses and notices are located at community.rti.com/documentation. IT IS YOUR RESPONSIBILITY TO ENSURE THAT YOUR USE OF THIRD-PARTY SOFTWARE COMPLIES WITH THE CORRESPONDING THIRD-PARTY LICENSE TERMS AND CONDITIONS.

Notices

Deprecations and Removals

Any deprecations or removals noted in this document serve as notice under the Real-Time Innovations, Inc. Maintenance Policy #4220 and/or any other agreements by and between RTI and customer regarding maintenance and support of RTI's software.

Deprecated means that the item is still supported in the release, but will be removed in a future release. *Removed* means that the item is discontinued or no longer supported. By specifying that an item is deprecated in a release, RTI hereby provides customer notice that RTI reserves the right after one year from the date of such release and, with or without further notice, to immediately terminate maintenance (including without limitation, providing updates and upgrades) for the item, and no longer support the item, in a future release.

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	
1.1 Paths Mentioned in Documentation	1
Chapter 2 Installing Queuing Service	
2.1 Installing on a Linux or macOS System	3
2.2 Installing on a Windows System	4
Chapter 3 Using the Examples	5
Chapter 4 Running Queuing Service	
4.1 Starting from Launcher	7
4.2 Starting Manually from the Command Line	7
4.3 Using Queuing Service as a Windows Service	10

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® Connex®*. The default installation paths are:

- macOS® systems:
/Applications/rti_connex_dds-7.3.0
- Linux systems, non-*root* user:
/home/<your user name>/rti_connex_dds-7.3.0
- Linux systems, *root* user:
/opt/rti_connex_dds-7.3.0

- Windows® systems, user without Administrator privileges:
`<your home directory>\rti_connex_dds-7.3.0`
- Windows systems, user with Administrator privileges:
`C:\Program Files\rti_connex_dds-7.3.0`

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_connex_dds-7.3.0\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/7.3.0/examples`
- Linux systems: `/home/<your user name>/rti_workspace/7.3.0/examples`
- Windows systems: `<your Windows documents folder>\rti_workspace\7.3.0\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 Connex 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 *Connex*. 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 *Connex*. 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 10](#)

If you are starting *Queuing Service* as a Windows Service, also read [4.3 Using Queuing Service as a Windows Service on page 10](#).

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/rtiqueuingervice [options]
```

Example:

```
cd <NDDSHOME>  
bin/rtiqueuingervice -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 <i>Queuing Service</i> . Remote commands will refer to the queuing service using this name. In addition, the name of <i>DomainParticipants</i> created by <i>Queuing Service</i> 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 from a File in the <i>Queuing Service User's Manual</i> .
-cfgName <name>	Specifies a configuration name. <i>Queuing Service</i> 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 <i>Queuing Service</i> will wait for an initial configuration when using -cfgRemote . Default: 20 seconds
-daemon	Runs <i>Queuing Service</i> as a daemon/Windows service. When this flag is present, <i>Queuing Service</i> 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 <i>Queuing Service</i> will use domains 50 and 51. Default: 0

^aThis 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
-heapSnapshotPeriod	<p>Enables heap monitoring.</p> <p><i>Queuing Service</i> will generate a heap snapshot every <sec>.</p> <p>Default: heap monitoring is disabled.</p>
-heapSnapshotDir	<p>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.</p> <p>Default: current working directory</p>
-help	Displays help information.
-remoteAdministrationDomainId <ID>	<p>Enables remote administration and sets the domain ID for remote communication.</p> <p>When remote administration is enabled, <i>Queuing Service</i> will create a <i>DomainParticipant</i>, <i>Publisher</i>, <i>Subscriber</i>, <i>DataWriter</i>, and <i>DataReader</i> in the designated domain.</p> <p>See Chapter 5, Administering Queuing Service from a Remote Location, in the <i>Queuing Service User's Manual</i>.</p> <p>This option overrides the value of the tag <domain_id> within a <administration> tag.</p> <p>This parameter is required when using -cfgRemote.</p> <p>Default: Remote administration is not enabled unless it is enabled from the XML file.</p>
-persistentFilePrefix	<p>Specifies a name prefix to use with all files created by <i>Queuing Service</i>.</p> <p>This option overrides the value of the tag <file_prefix> within <persistence_settings>/<filesystem>.</p> <p>Default: Value in <persistence_settings>/<filesystem>/<file_prefix>.</p>
-persistentStoragePath	<p>Configures the directory for persistent storage.</p> <p>This option overrides the value of the tag <directory> within <persistence_settings>/<filesystem>.</p> <p>Default: Value in <persistence_settings>/<filesystem>/<directory>.</p>
-var <name>=<value>	<p>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 <i>Queuing Service User's Manual</i> for more information on configuration variables.</p> <p>You may have more than one -var flag on the command line.</p> <p>On Windows platforms, you will need to put quotation marks around the variable name and value, like this:</p> <p>-var "MY_VAR=myvalue"</p>
-verbosity <n>	<p>Controls what type of messages are logged:</p> <ul style="list-style-type: none"> 0 - Silent 1 - Exceptions (<i>Connex</i> and <i>Queuing Service</i>) (default) 2 - Warnings (<i>Queuing Service</i>) 3 - Information (<i>Queuing Service</i>) 4 - Warnings (<i>Connex</i> and <i>Queuing Service</i>) 5 - Tracing (<i>Queuing Service</i>) 6 - Tracing (<i>Connex</i> and <i>Queuing Service</i>) <p>Each verbosity level, <i>n</i>, includes all the verbosity levels smaller than <i>n</i>.</p>
-version	Prints the <i>Queuing Service</i> 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 Connex 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 Connex Getting Started Guide](#).