



CoreDX™ Data Distribution Service  
The leading Small Footprint DDS Middleware

# C# Reference Manual

Twin Oaks Computing, Inc  
Castle Rock, CO 80108

Nov 2011

# TWINOAKS<sup>TM</sup> COMPUTING INC.

PRACTICAL MIDDLEWARE EXPERTISE

©2009-2011 Twin Oaks Computing, Inc

All rights reserved.

Published online 2009-2011

## Trademarks

Twin Oaks Computing, and CoreDX DDS, and the CoreDX DDS logo are trademarks of Twin Oaks Computing, Inc. All other products or company names mentioned are used for identification purposes only, and may be trademarks of their respective owners.

## Copy and Use Restrictions

No part of this document may be reproduced, stored, or transmitted (electronically or mechanically) without the prior written permission of Twin Oaks Computing, Inc. The software documented in this publication is provided pursuant to a License Agreement containing restrictions on its use.

## DISCLAIMER OF WARRANTY

THIS DOCUMENT IS PROVIDED "AS IS" AND ALL EXPRESS OR IMPLIED CONDITIONS, REPRESENTATIONS AND WARRANTIES, INCLUDING ANY IMPLIED WARRANTY OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE OR NON-INFRINGEMENT, ARE DISCLAIMED, EXCEPT TO THE EXTENT THAT SUCH DISCLAIMERS ARE HELD TO BE LEGALLY INVALID.

## Contact

Twin Oaks Computing, Inc  
755 Maleta Ln, Ste 203  
Castle Rock, CO 80108  
(720) 733-7906  
[contact@twinoakscomputing.com](mailto:contact@twinoakscomputing.com)  
<http://www.twinoakscomputing.com>  
[http://twitter.com/CoreDX\\_DDS](http://twitter.com/CoreDX_DDS)

# Contents

<b>1</b>	<b>Overview</b>	<b>1</b>
<b>2</b>	<b>Data Structure Documentation</b>	<b>5</b>
2.1	DDS Entities . . . . .	5
2.2	DDS Quality of Service . . . . .	7
2.3	DDS Conditions, Listeners, and WaitSets . . . . .	8
2.4	DDS Listeners . . . . .	9
2.5	DDS Conditions . . . . .	10
2.6	DDS WaitSets . . . . .	11
2.7	DDS Status Structures . . . . .	12
<b>3</b>	<b>API Documentation</b>	<b>15</b>
3.1	Condition Class Reference . . . . .	15
3.2	ContentFilteredTopic Class Reference . . . . .	17
3.3	DataReader Class Reference . . . . .	20
3.4	DataReaderListener Class Reference . . . . .	26
3.5	DataReaderQos Class Reference . . . . .	28
3.6	DataWriter Class Reference . . . . .	31
3.7	DataWriterListener Class Reference . . . . .	35
3.8	DataWriterQos Class Reference . . . . .	36
3.9	DDS Class Reference . . . . .	39
3.10	DomainEntity Class Reference . . . . .	45
3.11	DomainParticipant Class Reference . . . . .	46

---

3.12 DomainParticipantFactory Class Reference . . . . .	56
3.13 DomainParticipantFactoryQos Class Reference . . . . .	59
3.14 DomainParticipantListener Class Reference . . . . .	60
3.15 DomainParticipantQos Class Reference . . . . .	62
3.16 Entity Class Reference . . . . .	63
3.17 GuardCondition Class Reference . . . . .	65
3.18 InconsistentTopicStatus Class Reference . . . . .	66
3.19 LivelinessChangedStatus Class Reference . . . . .	67
3.20 LivelinessLostStatus Class Reference . . . . .	68
3.21 OfferedDeadlineMissedStatus Class Reference . . . . .	69
3.22 OfferedIncompatibleQosStatus Class Reference . . . . .	70
3.23 PublicationMatchedStatus Class Reference . . . . .	71
3.24 Publisher Class Reference . . . . .	72
3.25 PublisherListener Class Reference . . . . .	77
3.26 PublisherQos Class Reference . . . . .	78
3.27 QueryCondition Class Reference . . . . .	80
3.28 ReadCondition Class Reference . . . . .	82
3.29 RequestedDeadlineMissedStatus Class Reference . . . . .	84
3.30 RequestedIncompatibleQosStatus Class Reference . . . . .	85
3.31 SampleInfo Class Reference . . . . .	86
3.32 SampleLostStatus Class Reference . . . . .	89
3.33 SampleRejectedStatus Class Reference . . . . .	90
3.34 StatusCondition Class Reference . . . . .	91
3.35 Subscriber Class Reference . . . . .	93
3.36 SubscriberListener Class Reference . . . . .	98
3.37 SubscriberQos Class Reference . . . . .	100
3.38 SubscriptionMatchedStatus Class Reference . . . . .	102
3.39 Topic Class Reference . . . . .	103
3.40 TopicDescription Interface Reference . . . . .	106
3.41 TopicListener Class Reference . . . . .	107
3.42 TopicQos Class Reference . . . . .	108

---

3.43 WaitSet Class Reference . . . . .	110
<b>4 Data Structure Index</b>	<b>113</b>
4.1 Class List . . . . .	113
<b>5 Not Yet Supported</b>	<b>117</b>



# Chapter 1

## Overview

Welcome to the CoreDX DDS for C# API documentation from Twin Oaks Computing, Inc.

### Introduction

CoreDX DDS is a small-footprint, high-performance communications middleware compliant with the OMG Data Distribution Service (DDS) standard. CoreDX DDS supports multiple hardware architectures and operating systems, and is intended to facilitate the development of robust, near real-time, highly distributed systems.

This is the **CoreDX DDS for C# Reference Manual**. It provides a detailed reference for the CoreDX Data Distribution Service implementation from Twin Oaks Computing, Inc. The manual includes documentation on all of the CoreDX DDS data types and Application Programming Interface (API) routines.

The **CoreDX DDS Programmers Guide** provides more information on using the CoreDX API and related tools to produce a complete DDS enabled application.

The CoreDX DDS software provides a high-throughput, standards compliant, data communications infrastructure. CoreDX DDS offers the tools you need to realize Open Architecture goals. Built with a focus on performance, the CoreDX DDS software delivers a quality implementation of the OMG Data Distribution Service (DDS) standard.

The CoreDX DDS software implements the essential Data-Centric Publish-Subscribe (DCPS) communications layer as documented in the OMG DDS Standard. This standalone package, provides everything needed to integrate QoS enabled, Publish-Subscribe messaging into an application. The core software is written in the C language, and is optimized to be small and fast. The core package includes C and C++ language bindings for application integration. This reference manual describes the CoreDX DDS "C#" language binding.

## Intended Audience

This document is intended for software developers who are integrating the CoreDX DDS software into their application(s). The reference manual assumes that the reader is competent in programming languages and software development concepts. CoreDX DDS supports multiple languages, and this reference manual focuses on the C# programming language.

## Contents

The reference documentation includes information on the following API constructs:

1. Entities. This includes the primary objects with which an application must interact to enable DDS publish-subscribe communications.
  - [DomainParticipantFactory](#)
  - [DomainParticipant](#)
  - [Topic](#)
  - [ContentFilteredTopic](#)
  - [MultiTopic](#)
  - [Publisher](#)
  - [Subscriber](#)
  - [DataReader](#)
  - [DataWriter](#)
2. Quality of Service. This section documents the Quality of Service (QoS) structures that configure the behavior of the CoreDX middleware.
  - [DomainParticipantFactoryQos](#)
  - [DomainParticipantQos](#)
  - [TopicQos](#)
  - [PublisherQos](#)
  - [SubscriberQos](#)
  - [DataReaderQos](#)
  - [DataWriterQos](#)
3. Listeners and Events. This section covers the various structures and concepts involved in delivering events to the application.
  - [DomainParticipantListener](#)
  - [TopicListener](#)



- [PublisherListener](#)
- [SubscriberListener](#)
- [DataReaderListener](#)
- [DataWriterListener](#)

4. Status. This section describes the types of status maintained by the infrastructure.

- [InconsistentTopicStatus](#)
- [LivelinessChangedStatus](#)
- [LivelinessLostStatus](#)
- [OfferedDeadlineMissedStatus](#)
- [OfferedIncompatibleQosStatus](#)
- [PublicationMatchedStatus](#)
- [RequestedDeadlineMissedStatus](#)
- [RequestedIncompatibleQosStatus](#)
- [SampleLostStatus](#)
- [SampleRejectedStatus](#)
- [SubscriptionMatchedStatus](#)

5. Miscellaneous. This section includes miscellaneous support objects.

- [Condition](#)
- [GuardCondition](#)
- [StatusCondition](#)
- [QueryCondition](#)
- [WaitSet](#)



## Chapter 2

# Data Structure Documentation

### 2.1 DDS Entities

#### Classes

- class [DomainParticipant](#)

*The [DomainParticipant](#) is used to configure, create and destroy [Publisher](#), [Subscriber](#) and [Topic](#) objects.*

- class [DomainParticipantFactory](#)

*[DomainParticipantFactory](#) constructs [DomainParticipants](#). The*

- class [DataReader](#)

*The [DataReader](#) entity allows the application to subscribe to and read data.*

- class [DataWriter](#)

*The [DataWriter](#) entity provides an interface for the application to publish (write) data. The [DataWriter](#) is an abstract class that is extended to support a particular data type required by the application. A [DataReader](#) is associated with, and writes on, a single [Topic](#).*

- class [Entity](#)

*Base class for all DDS Entities.*

- class [DomainEntity](#)

*Base class for all DDS Domain Entities.*

- class [Publisher](#)

*The [Publisher](#) configures, creates, manages and destroys [DataWriters](#).*

- class [Subscriber](#)

The [Subscriber](#) configures, creates, manages and destroys [DataReaders](#).

- class [Topic](#)

[Topic](#) is the basic description of data to be published or subscribed. A topic is identified by a **name** and a **type**. A [Topic](#) is created by calling [DomainParticipant.create\\_topic\(\)](#). Prior to creating a [Topic](#), the associated data type must be registered with the [DomainParticipant](#) via a call to the [TypeSupportXYZ.register\\_type\(\)](#) function. [The [register\\_type\(\)](#) function is auto-generated 'type-specific' code.]

- class [ContentFilteredTopic](#)

[ContentFilteredTopic](#) provides a topic that may include data filtered from a related [Topic](#). The [ContentFilteredTopic](#) is associated with another un-filtered topic **related\_topic**. It applies a filter to the data of the related topic. If a data sample passes the filter, it will be made available to a [DataReader](#) associated with the [ContentFilteredTopic](#).

- struct [DDS\\_MultiTopic](#)

[DDS\\_MultiTopic](#) provides a topic that may include data from multiple [Topics](#).

## 2.2 DDS Quality of Service

### Classes

- class [DomainParticipantFactoryQos](#)  
*Structure that holds [DomainParticipantFactory](#) Quality of Service policies.*
- class [DomainParticipantQos](#)  
*Structure that holds [DomainParticipant](#) Quality of Service policies.*
- class [TopicQos](#)  
*Structure that holds [DDS\\_Topic](#) Quality of Service policies.*
- class [PublisherQos](#)  
*Structure that holds [Publisher](#) Quality of Service policies.*
- class [SubscriberQos](#)  
*Structure that holds [DDS\\_Subscriber](#) Quality of Service policies.*
- class [DataWriterQos](#)  
*Structure that holds [DataWriter](#) Quality of Service policies.*
- class [DataReaderQos](#)  
*Structure that holds [DataReader](#) Quality of Service policies.*

## 2.3 DDS Conditions, Listeners, and WaitSets

### Modules

- [DDS Listeners](#)
- [DDS Conditions](#)
- [DDS WaitSets](#)

## 2.4 DDS Listeners

### Classes

- class [DomainParticipantListener](#)  
*The [DomainParticipantListener](#) provides asynchronous notification of [DomainParticipant](#) events.*
- class [TopicListener](#)  
*The [TopicListener](#) provides asynchronous notification of [Topic](#) events.*
- class [PublisherListener](#)  
*The [PublisherListener](#) provides asynchronous notification of [Publisher](#) events.*
- class [DataWriterListener](#)  
*The [DataWriterListener](#) provides asynchronous notification of [DataWriter](#) events.*
- class [SubscriberListener](#)  
*The [SubscriberListener](#) provides asynchronous notification of [Subscriber](#) events.*
- class [DataReaderListener](#)  
*The [DataReaderListener](#) provides asynchronous notification of [DataReader](#) events.*

## 2.5 DDS Conditions

### Classes

- class [Condition](#)

*A [Condition](#) can be added to a [WaitSet](#) to provide synchronous event notification.*

- class [GuardCondition](#)

*A [GuardCondition](#) is a [Condition](#) where the **trigger\_value** is under application control.*

- class [ReadCondition](#)

*A [ReadCondition](#) is a specialized [Condition](#) associated with a [DataReader](#).*

- class [QueryCondition](#)

*The **trigger\_value** is driven by the data available, after applying the filter, in the associated [DataReader](#).*



---

## 2.6 DDS WaitSets

### Classes

- class [WaitSet](#)

*A `DDS_WaitSet` maintains a set of [Condition](#) objects and allows the application to wait until one or more of them have a `trigger_value` of `TRUE`.*

## 2.7 DDS Status Structures

### Classes

- class [InconsistentTopicStatus](#)  
*Status related to the `on_inconsistent_topic` listener methods of the `TopicListener` structure.*
- class [OfferedDeadlineMissedStatus](#)  
*Status related to the `on_offered_deadline_missed` listener methods of the `DataWriter`, `Publisher`, and `DomainParticipant` structures.*
- class [OfferedIncompatibleQosStatus](#)  
*Status related to the `on_offered_incompatible_qos` listener methods of the `DataWriter`, `Publisher`, and `DomainParticipant` structures.*
- class [LivelinessLostStatus](#)  
*Status related to the `on_liveliness_lost` listener methods of the `DataWriter`, `Publisher`, and `DomainParticipant` structures.*
- class [PublicationMatchedStatus](#)  
*Status related to the `on_publication_matched` listener methods of the `DataWriter`, `Publisher`, and `DomainParticipant` structures.*
- class [RequestedDeadlineMissedStatus](#)  
*Status related to the `on_requested_deadline_missed` listener methods of the `DataReader`, `Subscriber`, and `DomainParticipant` structures.*
- class [RequestedIncompatibleQosStatus](#)  
*Status related to the `on_requested_incompatible_qos` listener methods of the `DataReader`, `Subscriber`, and `DomainParticipant` structures.*
- class [SampleRejectedStatus](#)  
*Status related to the `on_sample_rejected` listener methods of the `DataReader`, `Subscriber`, and `DomainParticipant` structures.*
- class [LivelinessChangedStatus](#)  
*Status related to the `on_liveliness_changed` listener methods of the `DataReader`, `Subscriber`, and `DomainParticipant` structures.*
- class [SubscriptionMatchedStatus](#)  
*Status related to the `on_subscription_matched` listener methods of the `DataReader`, `Subscriber`, and `DomainParticipant` structures.*
- class [SampleLostStatus](#)

Status related to the `on_sample_lost` listener methods of the *DataReader*, *Subscriber*, and *DomainParticipant* structures.

## Enumerations

- enum `SampleRejectedStatusKind` { `NOT_REJECTED`, `REJECTED_BY_INSTANCE_LIMIT`, `REJECTED_BY_SAMPLES_LIMIT`, `REJECTED_BY_SAMPLES_PER_INSTANCE_LIMIT` }

### 2.7.1 Detailed Description

- `InconsistentTopicStatus`
- `LivelinessChangedStatus`
- `LivelinessLostStatus`
- `OfferedDeadlineMissedStatus`
- `OfferedIncompatibleQosStatus`
- `PublicationMatchedStatus`
- `RequestedDeadlineMissedStatus`
- `RequestedIncompatibleQosStatus`
- `SampleLostStatus`
- `SampleRejectedStatus`
- `_SubscriptionMatchedStatus`

### 2.7.2 Enumeration Type Documentation

#### 2.7.2.1 enum `SampleRejectedStatusKind`

Enumerator:

*NOT\_REJECTED*

*REJECTED\_BY\_INSTANCE\_LIMIT*

*REJECTED\_BY\_SAMPLES\_LIMIT*

*REJECTED\_BY\_SAMPLES\_PER\_INSTANCE\_LIMIT*



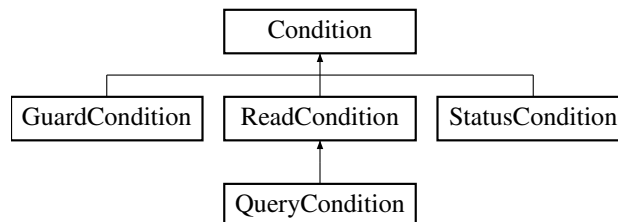
## Chapter 3

# API Documentation

### 3.1 Condition Class Reference

A [Condition](#) can be added to a [WaitSet](#) to provide synchronous event notification.

Inheritance diagram for Condition:



#### Public Member Functions

- `bool get\_trigger\_value ()`

#### 3.1.1 Detailed Description

A [Condition](#) can be added to a [WaitSet](#) to provide synchronous event notification. A [Condition](#) has a **trigger\_value** which can be true or false.

## 3.1.2 Member Function Documentation

### 3.1.2.1 `bool get_trigger_value ( ) [inline]`

This routine returns the current value of the **trigger\_value** in [Condition c](#).

A non-zero return value indicates that the **trigger\_value** is TRUE.

A zero return value indicates that the **trigger\_value** is FALSE.

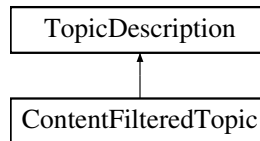
The documentation for this class was generated from the following file:

- `/home/ctucker/coredx_v3.4rc/src/dds_csharp/cond.cs`

## 3.2 ContentFilteredTopic Class Reference

[ContentFilteredTopic](#) provides a topic that may include data filtered from a related [Topic](#). The [ContentFilteredTopic](#) is associated with another un-filtered topic **related\_topic**. It applies a filter to the data of the related topic. If a data sample passes the filter, it will be made available to a [DataReader](#) associated with the [ContentFilteredTopic](#).

Inheritance diagram for ContentFilteredTopic:



### Public Member Functions

- [DomainParticipant](#) `get_participant ()`  
*This operation returns the parent [DomainParticipant](#) of the [Topic](#).*
- [String](#) `get_type_name ()`  
*This operation returns [type\\_name](#) of the [Topic](#).*
- [String](#) `get_name ()`  
*This operation returns [topic\\_name](#) of the [Topic](#).*
- [Topic](#) `get_related_topic ()`  
*This returns the real [Topic](#) associated with the [ContentFilteredTopic](#).*
- [ReturnCode\\_t](#) `get_expression_parameters (List< String > eparams)`  
*This accesses the current set of parameters used by the [ContentFilteredTopic](#).*
- [ReturnCode\\_t](#) `set_expression_parameters (List< String > filter_parameters)`  
*This specifies a new set of parameters for use with the [filter\\_expression](#).*

### 3.2.1 Detailed Description

[ContentFilteredTopic](#) provides a topic that may include data filtered from a related [Topic](#). The [ContentFilteredTopic](#) is associated with another un-filtered topic **related\_topic**. It applies a filter to the data of the related topic. If a data sample passes the filter, it will be made available to a [DataReader](#) associated with the [ContentFilteredTopic](#). The **filter\_expression** is an SQL like condition expression, and **filter\_parameters**

provide optional parameters that are referenced by the **filter\_expression**. The syntax of the filter expression is similar to the WHERE clause in SQL. For example "x<4" is a valid filter expression. It would test that data member 'x' is less than value '4'. If the filter expression evaluates to TRUE, then the data sample will be available, otherwise the data sample would be 'filtered' (excluded).

CoreDX DDS supports the 'LIKE' operator (and NOT LIKE) for regular expression string matching. The pattern string in a LIKE clause can contain "\*" to match zero or more characters, '\_' to match a single character, or '['<characters>']' to match a range of characters.

CoreDX DDS also includes support for the 'IN' operator. This provides a very powerful mechanism for testing that a value appears in a set of values. For example "symbol IN ('ge', 'msft', 'ibm')" will select all samples that have a symbol value of 'ge', 'msft', or 'ibm'. This could also be written as a series of equality tests combined with the OR operator; however, the IN operator is much more efficient. A filter that matches on several hundred or even thousands of values can be implemented very efficiently using the 'IN' operator.

The filter\_expression can refer to parameters. The syntax for parameters is the percent sign "%" followed by a number. The number is the index of the parameter in the **filter\_parameters** sequence. Parameters are counted starting at zero. So, "%0" refers to the first parameter, and "%4" refers to the fifth parameter. Using this syntax, the expression "x<%0" would test the value of 'x' against the first parameter in the sequence.

#### See also

DDS\_DomainParticipant\_create\_contentfilteredtopic()

## 3.2.2 Member Function Documentation

### 3.2.2.1 ReturnCode\_t get\_expression\_parameters ( List< String > *eparams* ) [inline]

This accesses the current set of parameters used by the [ContentFilteredTopic](#).

The **parameters** String Sequence is populated with the current set of parameters.

### 3.2.2.2 Topic get\_related\_topic ( ) [inline]

This returns the real [Topic](#) associated with the [ContentFilteredTopic](#).

That is, the [Topic](#) provided when the [ContentFilteredTopic](#) was created.

### 3.2.2.3 ReturnCode\_t set\_expression\_parameters ( List< String > *filter\_parameters* ) [inline]

This specifies a new set of parameters for use with the filter\_expression.

The **filter\_expression** is an SQL like condition expression, and the **parameters** argument provides optional parameters that are referenced by the **filter\_expression**. The syntax for referring to parameters in a filter\_expression is the percent sign "%" followed by a number. The number is the index of the parameter in the **filter\_parameters** sequence. Parameters are counted starting at zero. So, "%0" refers to the first parameter,



and "%4" refers to the fifth parameter. Using this syntax, the expression "x<%0" would test the value of 'x' against the first parameter in the sequence.

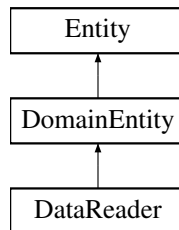
The documentation for this class was generated from the following file:

- /home/ctucker/coredx\_v3.4rc/src/dds\_csharp/top.cs

### 3.3 DataReader Class Reference

The [DataReader](#) entity allows the application to subscribe to and read data.

Inheritance diagram for DataReader:



#### Public Member Functions

- new [ReturnCode\\_t enable](#) ()
- override [InstanceHandle\\_t get\\_instance\\_handle](#) ()
- [ReadCondition create\\_readcondition](#) (uint sample\_states, uint view\_states, uint instance\_states)
- [QueryCondition create\\_querycondition](#) (uint sample\_states, uint view\_states, uint instance\_states, String query\_expression, List< String > query\_parameters)
  - Creates a DDS\_QueryCondition.*
- [ReturnCode\\_t delete\\_readcondition](#) ([ReadCondition rc](#))
- [ReturnCode\\_t delete\\_contained\\_entities](#) ()
- [ReturnCode\\_t set\\_qos](#) ([DataReaderQos qos](#))
- [ReturnCode\\_t get\\_qos](#) ([DataReaderQos qos](#))
- [ReturnCode\\_t set\\_listener](#) ([DataReaderListener new\\_listener](#), uint mask)
- [DataReaderListener get\\_listener](#) ()
- [TopicDescription get\\_topicdescription](#) ()
- [Subscriber get\\_subscriber](#) ()
- [ReturnCode\\_t get\\_sample\\_rejected\\_status](#) (out [SampleRejectedStatus status](#))
- [ReturnCode\\_t get\\_liveliness\\_changed\\_status](#) (out [LivelinessChangedStatus status](#))
- [ReturnCode\\_t get\\_requested\\_deadline\\_missed\\_status](#) (out [RequestedDeadlineMissedStatus status](#))
- [ReturnCode\\_t get\\_requested\\_incompatible\\_qos\\_status](#) (out [RequestedIncompatibleQosStatus status](#))
- [ReturnCode\\_t get\\_subscription\\_matched\\_status](#) (out [SubscriptionMatchedStatus status](#))
- [ReturnCode\\_t get\\_sample\\_lost\\_status](#) (out [SampleLostStatus status](#))
- [ReturnCode\\_t wait\\_for\\_historical\\_data](#) ([Duration\\_t max\\_wait](#))
- [ReturnCode\\_t get\\_matched\\_publications](#) (List< [InstanceHandle\\_t](#) > publication\_handles)

### 3.3.1 Detailed Description

The [DataReader](#) entity allows the application to subscribe to and read data. The [DataReadre](#) is an abstract class that is extended to support a particular data type required by the application. A [DataReader](#) is associated with a single [TopicDescription](#) ([Topic](#), [MultiTopic](#), or [ContentFilteredTopic](#)).

### 3.3.2 Member Function Documentation

#### 3.3.2.1 [QueryCondition](#) `create_querycondition ( uint sample_states, uint view_states, uint instance_states, String query_expression, List< String > query_parameters ) [inline]`

Creates a [DDS\\_QueryCondition](#).

The returned [QueryCondition](#) can be used as an argument to `read_w_condition()` or `take_w_condition()`.

The **query\_expression** is an SQL like condition expression, and **query\_parameters** provide optional parameters that are referenced by the **query\_expression**. The syntax of the query expression is similar to the WHERE clause in SQL. For example "x<4" is a valid query expression. It would test that data member 'x' is less than value '4'. If the query expression evaluates to TRUE, then the data sample will be available, otherwise the data sample will be 'filtered' (excluded).

CoreDX [DDS](#) supports the '**LIKE**' operator (and NOT LIKE) for regular expression string matching. The pattern string in a LIKE clause can contain '\*' to match zero or more characters, '\_' to match a single character, or '['<characters>']' to match a range of characters.

CoreDX [DDS](#) also includes support for the '**IN**' operator. This provides a very powerful mechanism for testing that a value appears in a set of values. For example "symbol IN ('ge', 'msft', 'ibm')" will select all samples that have a symbol value of 'ge', 'msft', or 'ibm'. This could also be written as a series of equality tests combined with the OR operator; however, the IN operator is much more efficient. A query that matches on several hundred or even thousands of values can be implemented very efficiently using the 'IN' operator.

The query\_expression can refer to parameters. The syntax for paramters is the percent sign '%' followed by a number. The number is the index of the paramter in the **query\_paramters** sequence. Parameters are counted starting at zero. So, "%0" refers to the first parameter, and "%4" refers to the fifth paramter. Using this syntax, the expression "x<%0" would test the value of 'x' against the first parameter in the sequence.

#### See also

`FooDataReader::read_w_condition()`  
`FooDataReader::take_w_condition()`

#### Not Yet Supported

[QueryConditions](#) are not yet supported as triggers for a [WaitSet](#).

### 3.3.2.2 `ReadCondition create_readcondition ( uint sample_states, uint view_states, uint instance_states ) [inline]`

Creates a [ReadCondition](#) that is associated with this [DataReader](#). The returned condition can be added to a [WaitSet](#) or used in a call to the specialized `read()` or `take()` operations. For example see `DataReaderFoo.read_w_condition()`;

### 3.3.2.3 `ReturnCode_t delete_contained_entities ( ) [inline]`

This operation deletes all the [ReadCondition](#) and [QueryCondition](#) objects previously created by means of the `DataReader.create_readcondition()` and `DataReader.create_querycondition()` operations.

After successful execution, the application may delete the [Publisher](#) by calling `Subscriber.delete_datareader()`.

If any of the objects cannot be deleted, this routine will return `ReturnCode_t.RETCODE_PRECONDITION_NOT_MET`.

### 3.3.2.4 `ReturnCode_t delete_readcondition ( ReadCondition rc ) [inline]`

Destroys a [ReadCondition](#) (or [QueryCondition](#)). The provided `a_condition` must have been previously created via a call to `DataReader.create_readcondition()` or `DataReader.create_querycondition()`.

The `a_condition` argument must belong to [DataReader dr](#). Otherwise, the error `ReturnCode_t.RETCODE_PRECONDITION_NOT_MET` will be returned.

If the [DataReader](#) is actively processing the [ReadCondition](#), this routine will return `ReturnCode_t.RETCODE_ERROR`; in this case, the `delete_readcondition()` call should be re-tried.

### 3.3.2.5 `new ReturnCode_t enable ( ) [inline, virtual]`

Enables the [DataReader](#). A [DataReader](#) is created either enabled or not based on the [SubscriberQos](#) setting `entity_factory`. When a [DataReader](#) is not enabled, only the following sub-set of all [DataReader](#) operations are legal:

- operations to get and set QoS policies,
- `get_statuscondition()`,
- `get_status_changes()`,

Any other operation may return the `ReturnCode_t.RETCODE_NOT_ENABLED` error. `DataReader_enable()` may be called on an already enabled [DataReader](#) [it will have no effect].

Reimplemented from [Entity](#).

**3.3.2.6** `override InstanceHandle_t get_instance_handle ( ) [inline, virtual]`

Gets the handle that locally identifies this [Entity](#).

Reimplemented from [Entity](#).

**3.3.2.7** `DataReaderListener get_listener ( ) [inline]`

This operation returns the currently installed [DataReaderListener](#).

**3.3.2.8** `ReturnCode_t get_liveliness_changed_status ( out LivelinessChangedStatus status ) [inline]`

Provides access to the current [LivelinessChangedStatus](#) of the [DataReader](#). As a side-effect, this routine will reset the `total_count_change` status field to zero.

**3.3.2.9** `ReturnCode_t get_matched_publications ( List< InstanceHandle_t > publication_handles ) [inline]`

This operation retrieves the list of DataWriters currently matched with this [DataReader dr](#). This list will include the handles that identify DataWriters which have matching [Topic](#) and compatible QoS with [DataReader](#).

If a [DataWriter](#) has been ignored by a call to [DomainParticipant.ignore\\_publication\(\)](#), then it will not appear in the list.

**Parameters**

*publication\_handles* A vector that will be populated with InstanceHandle\_t(s).

**3.3.2.10** `ReturnCode_t get_qos ( DataReaderQos qos ) [inline]`

Returns the current [DataReaderQos](#) settings held in the [DataReader dr](#). This routines copies data from the [DataReader](#) QoS properties into `qos`.

**3.3.2.11** `ReturnCode_t get_requested_deadline_missed_status ( out RequestedDeadlineMissedStatus status ) [inline]`

Provides access to the current [RequestedDeadlineMissedStatus](#) of the [DataReader](#). As a side-effect, this routine will reset the `total_count_change` status field to zero.

### 3.3.2.12 `ReturnCode_t get_requested_incompatible_qos_status ( out RequestedIncompatibleQosStatus status ) [inline]`

Provides access to the current [RequestedIncompatibleQosStatus](#) of the [DataReader](#). As a side-effect, this routine will reset the `total_count_change` status field to zero.

### 3.3.2.13 `ReturnCode_t get_sample_lost_status ( out SampleLostStatus status ) [inline]`

Provides access to the current [SampleLostStatus](#) of the [DataReader](#). As a side-effect, this routine will reset the `total_count_change` status field to zero.

### 3.3.2.14 `ReturnCode_t get_sample_rejected_status ( out SampleRejectedStatus status ) [inline]`

Provides access to the current [SampleRejectedStatus](#) of the [DataReader](#). As a side-effect, this routine will reset the `total_count_change` status field to zero.

### 3.3.2.15 `Subscriber get_subscriber ( ) [inline]`

Returns the Subscriber that contains [DataReader dr](#).

### 3.3.2.16 `ReturnCode_t get_subscription_matched_status ( out SubscriptionMatchedStatus status ) [inline]`

Provides access to the current [SubscriptionMatchedStatus](#) of the [DataReader](#). As a side-effect, this routine will reset the `total_count_change` status field to zero.

### 3.3.2.17 `TopicDescription get_topicdescription ( ) [inline]`

Returns the [TopicDescription](#) associated with [DataReader dr](#).

### 3.3.2.18 `ReturnCode_t set_listener ( DataReaderListener new_listener, uint mask ) [inline]`

Installs a [DataReaderListener](#) on [DataReader dr](#). Only one listener may be attached to a [DataReader](#) at a time. A call to `set_listener()` will replace any current listener with `a_listener`.

`a_listener` can be NULL, which indicates a listener that does nothing.

### 3.3.2.19 ReturnCode\_t set\_qos ( DataReaderQos qos ) [inline]

Sets the [DataReaderQos](#) values. These QoS values affect the behavior of the [DataReader](#). This routine may fail if the provided `qos` argument is not internally consistent. In this case, `ReturnCode_t.RETCODE_INCONSISTENT_POLICY` will be returned, and no changes will be made to the [DataReader](#) QoS.

### 3.3.2.20 ReturnCode\_t wait\_for\_historical\_data ( Duration\_t max\_wait ) [inline]

This routine blocks until all 'historical' data is received.

#### Parameters

*max\_wait* The maximum amount of time to block while waiting. Can be set to { `INFINITE_SEC`, `INFINITE_NSEC` }

The documentation for this class was generated from the following file:

- `/home/ctucker/coredx_v3.4rc/src/dds_csharp/dr.cs`

## 3.4 DataReaderListener Class Reference

The [DataReaderListener](#) provides asynchronous notification of [DataReader](#) events.

### Public Attributes

- [requested\\_deadline\\_missed\\_delegate](#) [on\\_requested\\_deadline\\_missed](#)
- [requested\\_incompatible\\_qos\\_delegate](#) [on\\_requested\\_incompatible\\_qos](#)
- [sample\\_rejected\\_delegate](#) [on\\_sample\\_rejected](#)
- [liveliness\\_changed\\_delegate](#) [on\\_liveliness\\_changed](#)
- [data\\_available\\_delegate](#) [on\\_data\\_available](#)
- [subscription\\_matched\\_delegate](#) [on\\_subscription\\_matched](#)
- [sample\\_lost\\_delegate](#) [on\\_sample\\_lost](#)

### 3.4.1 Detailed Description

The [DataReaderListener](#) provides asynchronous notification of [DataReader](#) events. This listener can be installed during [DataReader](#) creation, [Subscriber.create\\_datareader\(\)](#), as well as by calling [DataReader.set\\_listener\(\)](#).

#### Note

The listener callback methods should be lightweight and should not block. If a callback method blocks, it will block all other callback operations within the same [DomainParticipant](#).

### 3.4.2 Member Data Documentation

**3.4.2.1** [data\\_available\\_delegate](#) [on\\_data\\_available](#)

**3.4.2.2** [liveliness\\_changed\\_delegate](#) [on\\_liveliness\\_changed](#)

**3.4.2.3** [requested\\_deadline\\_missed\\_delegate](#) [on\\_requested\\_deadline\\_missed](#)

**3.4.2.4** [requested\\_incompatible\\_qos\\_delegate](#) [on\\_requested\\_incompatible\\_qos](#)

**3.4.2.5** [sample\\_lost\\_delegate](#) [on\\_sample\\_lost](#)

**3.4.2.6** [sample\\_rejected\\_delegate](#) [on\\_sample\\_rejected](#)

**3.4.2.7** [subscription\\_matched\\_delegate](#) [on\\_subscription\\_matched](#)

The documentation for this class was generated from the following file:



- `/home/ctucker/coredx_v3.4rc/src/dds_csharp/base.cs`

## 3.5 DataReaderQos Class Reference

Structure that holds [DataReader](#) Quality of Service policies.

### Public Attributes

- [DurabilityQosPolicy](#) durability
- [DeadlineQosPolicy](#) deadline
- [LatencyBudgetQosPolicy](#) latency\_budget
- [LivelinessQosPolicy](#) liveliness
- [ReliabilityQosPolicy](#) reliability
- [DestinationOrderQosPolicy](#) destination\_order
- [HistoryQosPolicy](#) history
- [ResourceLimitsQosPolicy](#) resource\_limits
- [UserDataQosPolicy](#) user\_data
- [OwnershipQosPolicy](#) ownership
- [TimeBasedFilterQosPolicy](#) time\_based\_filter
- [ReaderDataLifecycleQosPolicy](#) reader\_data\_lifecycle

### 3.5.1 Detailed Description

Structure that holds [DataReader](#) Quality of Service policies.

#### See also

[DataReader::set\\_qos\(\)](#)  
[DataReader::get\\_qos\(\)](#)  
[Subscriber::create\\_datareader\(\)](#)  
[Subscriber::set\\_default\\_datareader\\_qos\(\)](#)  
[Subscriber::get\\_default\\_datareader\\_qos\(\)](#)

### 3.5.2 Member Data Documentation

#### 3.5.2.1 DeadlineQosPolicy deadline

The requested update frequency for data instances.

#### 3.5.2.2 DestinationOrderQosPolicy destination\_order

The destination order logic requested by the [DataReader](#).

**3.5.2.3 DurabilityQosPolicy durability**

The durability policy requested by the [DataReader](#).

**3.5.2.4 HistoryQosPolicy history**

The data history requested by the [DataReader](#).

**3.5.2.5 LatencyBudgetQosPolicy latency\_budget**

The latency requested by the [DataReader](#).

**3.5.2.6 LivelinessQosPolicy liveliness**

The liveliness mechanism requested by the [DataReader](#).

**3.5.2.7 OwnershipQosPolicy ownership**

The type of 'ownership' offered by the [DataReader](#).

**3.5.2.8 ReaderDataLifecycleQosPolicy reader\_data\_lifecycle**

Controls the auto-purge behavior of the [DataReader](#).

**3.5.2.9 ReliabilityQosPolicy reliability**

The transport reliability requested by the [DataReader](#).

**3.5.2.10 ResourceLimitsQosPolicy resource\_limits**

The resource limits set on the [DataReader](#).

**3.5.2.11 TimeBasedFilterQosPolicy time\_based\_filter**

The maximum update frequency required/desired by the [DataReader](#).

### 3.5.2.12 UserDataQosPolicy user\_data

A sequence of octets associated with the [DataReader](#).

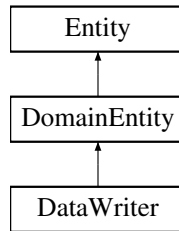
The documentation for this class was generated from the following file:

- /home/ctucker/coredx\_v3.4rc/src/dds\_csharp/base.cs

## 3.6 DataWriter Class Reference

The [DataWriter](#) entity provides an interface for the application to publish (write) data. The [DataWriter](#) is an abstract class that is extended to support a particular data type required by the application. A [DataReader](#) is associated with, and writes on, a single [Topic](#).

Inheritance diagram for DataWriter:



### Public Member Functions

- override [ReturnCode\\_t enable](#) ()
- override [InstanceHandle\\_t get\\_instance\\_handle](#) ()
- [ReturnCode\\_t set\\_qos](#) ([DataWriterQos](#) qos)
- [ReturnCode\\_t get\\_qos](#) ([DataWriterQos](#) qos)
- [ReturnCode\\_t set\\_listener](#) ([DataWriterListener](#) new\_listener, uint mask)
- [DataWriterListener get\\_listener](#) ()
- [Topic get\\_topic](#) ()
- [Publisher get\\_publisher](#) ()
- [ReturnCode\\_t wait\\_for\\_acknowledgments](#) ([Duration\\_t](#) max\_wait)
  - Block until this writer has received acknowledgements for all written data.*
- [ReturnCode\\_t assert\\_liveliness](#) ()
- [ReturnCode\\_t get\\_liveliness\\_lost\\_status](#) (out [LivelinessLostStatus](#) status)
- [ReturnCode\\_t get\\_offered\\_deadline\\_missed\\_status](#) (out [OfferedDeadlineMissedStatus](#) status)
- [ReturnCode\\_t get\\_offered\\_incompatible\\_qos\\_status](#) (out [OfferedIncompatibleQosStatus](#) status)
- [ReturnCode\\_t get\\_publication\\_matched\\_status](#) (out [PublicationMatchedStatus](#) status)
- [ReturnCode\\_t get\\_matched\\_subscriptions](#) (List< [InstanceHandle\\_t](#) > subscription\_handles)
- [ReturnCode\\_t get\\_matched\\_subscription\\_data](#) ([SubscriptionBuiltinTopicData](#) subscription\_data, [InstanceHandle\\_t](#) subscription\_handle)

### 3.6.1 Detailed Description

The [DataWriter](#) entity provides an interface for the application to publish (write) data. The [DataWriter](#) is an abstract class that is extended to support a particular data type required by the application. A [DataReader](#) is associated with, and writes on, a single [Topic](#).

## 3.6.2 Member Function Documentation

### 3.6.2.1 ReturnCode\_t assert\_liveliness ( ) [inline]

This operation manually asserts the liveliness of the [DataWriter](#) **dw**. This operation is useful if the LIVE-LINESS QoS setting is MANUAL\_BY\_PARTICIPANT\_LIVELINESS\_QOS or MANUAL\_BY\_TOPIC\_LIVELINESS\_QOS; otherwise, it has no effect.

The **write** operation automatically asserts liveliness on the [DataWriter](#) and its [DomainParticipant](#). Therefore, **assert\_liveliness** is required only if the application is not writing data frequently enough to satisfy the LIVELINESS setting.

### 3.6.2.2 override ReturnCode\_t enable ( ) [inline, virtual]

Enables the [DataWriter](#). A [DataWriter](#) is created either enabled or not based on the [PublisherQos](#) setting **entity\_factory**. When a [DataWriter](#) is not enabled, only the following sub-set of all [DataWriter](#) operations are legal:

- operations to get and set QoS policies,
- [get\\_statuscondition\(\)](#),
- [get\\_status\\_changes\(\)](#),

Any other operation may return the RETCODE\_NOT\_ENABLED error. [DataWriter\\_enable\(\)](#) may be called on an already enabled [DataWriter](#) [it will have no effect].

Reimplemented from [Entity](#).

### 3.6.2.3 override InstanceHandle\_t get\_instance\_handle ( ) [inline, virtual]

Gets the handle that locally identifies this [Entity](#).

Reimplemented from [Entity](#).

### 3.6.2.4 DataWriterListener get\_listener ( ) [inline]

This operation returns the currently installed [DataWriterListener](#).

### 3.6.2.5 ReturnCode\_t get\_liveliness\_lost\_status ( out LivelinessLostStatus *status* ) [inline]

Provides access to the current [LivelinessLostStatus](#) of the [DataWriter](#). As a side-effect, this routine will reset the **total\_count\_change** status field to zero.

### 3.6.2.6 `ReturnCode_t get_matched_subscription_data ( SubscriptionBuiltinTopicData subscription_data, InstanceHandle_t subscription_handle ) [inline]`

This operation returns data that describes a particular matched [DataReader](#) identified by **subscription\_handle**. An appropriate handle can be obtained through a call to [DataWriter.get\\_matched\\_subscriptions\(\)](#).

If **subscription\_handle** does not identify a matched [DataReader](#), this routine will return `ReturnCode_t.RETCODE_PRECONDITION_NOT_MET`.

### 3.6.2.7 `ReturnCode_t get_matched_subscriptions ( List< InstanceHandle_t > subscription_handles ) [inline]`

This operation retrieves the list of [DataReaders](#) currently matched with the [DataWriter dw](#). This list will include the handles that identify [DataReaders](#) which have matching [Topic](#) and compatible QoS with [DataWriter](#).

If a [DataReader](#) has been ignored by a call to [DomainParticipant.ignore\\_subscription\(\)](#), then it will not appear in the list.

#### Parameters

*subscription\_handles* A vector that will be populated with `InstanceHandle_t(s)`.

### 3.6.2.8 `ReturnCode_t get_offered_deadline_missed_status ( out OfferedDeadlineMissedStatus status ) [inline]`

Provides access to the current [OfferedDeadlineMissedStatus](#) of the [DataWriter](#). As a side-effect, this routine will reset the **total\_count\_change** status field to zero.

### 3.6.2.9 `ReturnCode_t get_offered_incompatible_qos_status ( out OfferedIncompatibleQosStatus status ) [inline]`

Provides access to the current [OfferedIncompatibleQosStatus](#) of the [DataWriter](#). As a side-effect, this routine will reset the **total\_count\_change** status field to zero.

### 3.6.2.10 `ReturnCode_t get_publication_matched_status ( out PublicationMatchedStatus status ) [inline]`

Provides access to the current [PublicationMatchedStatus](#) of the [DataWriter](#). As a side-effect, this routine will reset the **total\_count\_change** and **current\_count\_change** status fields to zero.

### 3.6.2.11 `Publisher get_publisher ( ) [inline]`

Returns the [Publisher](#) that contains [DataWriter dw](#).

### 3.6.2.12 `ReturnCode_t get_qos ( DataWriterQos qos ) [inline]`

Returns the current `DataWriterQos` settings held in the `DataWriter dw`. This routine copies data from the `DataWriter` QoS properties into `qos`.

### 3.6.2.13 `Topic get_topic ( ) [inline]`

Returns the `Topic` associated with `DataWriter dw`.

### 3.6.2.14 `ReturnCode_t set_listener ( DataWriterListener new_listener, uint mask ) [inline]`

Installs a `DataWriterListener` on `DataWriter dw`. Only one listener may be attached to a `DataWriter` at a time. A call to `set_listener()` will replace any current listener with `a_listener`.

`a_listener` can be NULL, which indicates a listener that does nothing.

### 3.6.2.15 `ReturnCode_t set_qos ( DataWriterQos qos ) [inline]`

Sets the `DataWriterQos` values. These QoS values affect the behavior of the `DataWriter`. This routine may fail if the provided `qos` argument is not internally consistent. In this case, `ReturnCode_t.RETCODE_INCONSISTENT_POLICY` will be returned, and no changes will be made to the `DataWriter` QoS.

### 3.6.2.16 `ReturnCode_t wait_for_acknowledgments ( Duration_t max_wait ) [inline]`

Block until this writer has received acknowledgements for all written data.

This routine will block until all data written by the writer has been acknowledged, or until the 'max\_wait' duration has passed. 'max\_wait' can be set to INFINITE, in which case this routine may block indefinitely.

#### Return values

`DDS_RETCODE_TIME_OUT` returned if 'max\_wait' passes before all acks are received

`DDS_RETCODE_OK` returned if all acks have been received before 'max\_wait'

The documentation for this class was generated from the following file:

- /home/ctucker/coredx\_v3.4rc/src/dds\_csharp/dw.cs



## 3.7 DataWriterListener Class Reference

The [DataWriterListener](#) provides asynchronous notification of [DataWriter](#) events.

### Public Attributes

- [offered\\_deadline\\_missed\\_delegate](#) [on\\_offered\\_deadline\\_missed](#)
- [offered\\_incompatible\\_qos\\_delegate](#) [on\\_offered\\_incompatible\\_qos](#)
- [liveliness\\_lost\\_delegate](#) [on\\_liveliness\\_lost](#)
- [publication\\_matched\\_delegate](#) [on\\_publication\\_matched](#)

### 3.7.1 Detailed Description

The [DataWriterListener](#) provides asynchronous notification of [DataWriter](#) events. This listener can be installed during [DataWriter](#) creation, [Publisher.create\\_datawriter\(\)](#), as well as by calling [DataWriter.set\\_listener\(\)](#).

#### Note

The listener callback methods should be lightweight and should not block. If a callback method blocks, it will block all other callback operations within the same [DomainParticipant](#).

### 3.7.2 Member Data Documentation

**3.7.2.1 [liveliness\\_lost\\_delegate](#) [on\\_liveliness\\_lost](#)**

**3.7.2.2 [offered\\_deadline\\_missed\\_delegate](#) [on\\_offered\\_deadline\\_missed](#)**

**3.7.2.3 [offered\\_incompatible\\_qos\\_delegate](#) [on\\_offered\\_incompatible\\_qos](#)**

**3.7.2.4 [publication\\_matched\\_delegate](#) [on\\_publication\\_matched](#)**

The documentation for this class was generated from the following file:

- `/home/ctucker/coredx_v3.4rc/src/dds_csharp/base.cs`

## 3.8 DataWriterQos Class Reference

Structure that holds [DataWriter](#) Quality of Service policies.

### Public Attributes

- [DurabilityQosPolicy](#) durability
- [DurabilityServiceQosPolicy](#) durability\_service
- [DeadlineQosPolicy](#) deadline
- [LatencyBudgetQosPolicy](#) latency\_budget
- [LivelinessQosPolicy](#) liveliness
- [ReliabilityQosPolicy](#) reliability
- [DestinationOrderQosPolicy](#) destination\_order
- [HistoryQosPolicy](#) history
- [ResourceLimitsQosPolicy](#) resource\_limits
- [TransportPriorityQosPolicy](#) transport\_priority
- [LifespanQosPolicy](#) lifespan
- [UserDataQosPolicy](#) user\_data
- [OwnershipQosPolicy](#) ownership
- [OwnershipStrengthQosPolicy](#) ownership\_strength
- [WriterDataLifecycleQosPolicy](#) writer\_data\_lifecycle

### 3.8.1 Detailed Description

Structure that holds [DataWriter](#) Quality of Service policies.

#### See also

[DataWriter::set\\_qos\(\)](#)  
[DataWriter::get\\_qos\(\)](#)  
[Publisher::create\\_datawriter\(\)](#)  
[Publisher::set\\_default\\_datawriter\\_qos\(\)](#)  
[Publisher::get\\_default\\_datawriter\\_qos\(\)](#)

### 3.8.2 Member Data Documentation

#### 3.8.2.1 DeadlineQosPolicy deadline

The deadline committed to by the [DataWriter](#).

#### 3.8.2.2 DestinationOrderQosPolicy destination\_order

The destination order logic offered by the [DataWriter](#).

### 3.8.2.3 DurabilityQosPolicy durability

The durability policy offered by the [DataWriter](#).

### 3.8.2.4 DurabilityServiceQosPolicy durability\_service

The durability service configuration offered by the [DataWriter](#).

### 3.8.2.5 HistoryQosPolicy history

The data history maintained by the [DataWriter](#).

### 3.8.2.6 LatencyBudgetQosPolicy latency\_budget

The latency allowed by the [DataWriter](#).

### 3.8.2.7 LifespanQosPolicy lifespan

The expiration time for old samples managed by the [DataWriter](#).

### 3.8.2.8 LivelinessQosPolicy liveliness

The liveliness mechanism offered by the [DataWriter](#).

### 3.8.2.9 OwnershipQosPolicy ownership

The type of 'ownership' offered by the [DataWriter](#).

### 3.8.2.10 OwnershipStrengthQosPolicy ownership\_strength

The measure of 'ownership strength' offered by the [DataWriter](#).

### 3.8.2.11 ReliabilityQosPolicy reliability

The transport reliability offered by the [DataWriter](#).

### 3.8.2.12 ResourceLimitsQosPolicy resource\_limits

The resource limits set on the [DataWriter](#).

### 3.8.2.13 TransportPriorityQosPolicy transport\_priority

The transport priority supported by the [DataWriter](#).

### 3.8.2.14 UserDataQosPolicy user\_data

A sequence of octets associated with the [DataWriter](#).

### 3.8.2.15 WriterDataLifecycleQosPolicy writer\_data\_lifecycle

Indicates if unregistered instances should be automatically disposed by the [DataWriter](#).

The documentation for this class was generated from the following file:

- `/home/ctucker/coredx_v3.4rc/src/dds_csharp/base.cs`

## 3.9 DDS Class Reference

The 'DDS' class includes several convenient constants.

### Public Attributes

- const int `LENGTH_UNLIMITED` = -1
- const int `DURATION_INFINITE_SEC` = 0x7fffffff
- const uint `DURATION_INFINITE_NSEC` = 0xffffffff
- const int `DURATION_ZERO_SEC` = 0
- const int `DURATION_ZERO_NSEC` = 0
- const int `TIMESTAMP_INVALID_SEC` = -1
- const uint `TIMESTAMP_INVALID_NSEC` = 0xffffffff
- const uint `READ_SAMPLE_STATE` = 0x0001
- const uint `NOT_READ_SAMPLE_STATE` = 0x0002
- const uint `ANY_SAMPLE_STATE` = 0x00FF
- const uint `NEW_VIEW_STATE` = 0x0001
- const uint `NOT_NEW_VIEW_STATE` = 0x0002
- const uint `ANY_VIEW_STATE` = 0x00FF
- const uint `ALIVE_INSTANCE_STATE` = 0x0001
- const uint `NOT_ALIVE_DISPOSED_INSTANCE_STATE` = 0x0002
- const uint `NOT_ALIVE_NO_WRITERS_INSTANCE_STATE` = 0x0004
- const uint `NOT_ALIVE_INSTANCE_STATE` = 0x0006
- const uint `ANY_INSTANCE_STATE` = 0x00FF
- const uint `INCONSISTENT_TOPIC_STATUS` = 0x0001
- const uint `OFFERED_DEADLINE_MISSED_STATUS` = 0x0002
- const uint `REQUESTED_DEADLINE_MISSED_STATUS` = 0x0004
- const uint `OFFERED_INCOMPATIBLE_QOS_STATUS` = 0x0020
- const uint `REQUESTED_INCOMPATIBLE_QOS_STATUS` = 0x0040
- const uint `SAMPLE_LOST_STATUS` = 0x0080
- const uint `SAMPLE_REJECTED_STATUS` = 0x0100
- const uint `DATA_ON_READERS_STATUS` = 0x0200
- const uint `DATA_AVAILABLE_STATUS` = 0x0400
- const uint `LIVELINESS_LOST_STATUS` = 0x0800
- const uint `LIVELINESS_CHANGED_STATUS` = 0x1000
- const uint `PUBLICATION_MATCHED_STATUS` = 0x2000
- const uint `SUBSCRIPTION_MATCHED_STATUS` = 0x4000
- const uint `ALL_STATUS` = 0xffff
- const `QosPolicyId_t` `USERDATA_QOS_POLICY_ID` = `QosPolicyId_t.USERDATA_QOS_POLICY_ID`
- const `QosPolicyId_t` `DURABILITY_QOS_POLICY_ID` = `QosPolicyId_t.DURABILITY_QOS_POLICY_ID`

- const `QosPolicyId_t PRESENTATION_QOS_POLICY_ID` = `QosPolicyId_t.PRESENTATION_QOS_POLICY_ID`
- const `QosPolicyId_t DEADLINE_QOS_POLICY_ID` = `QosPolicyId_t.DEADLINE_QOS_POLICY_ID`
- const `QosPolicyId_t LATENCYBUDGET_QOS_POLICY_ID` = `QosPolicyId_t.LATENCYBUDGET_QOS_POLICY_ID`
- const `QosPolicyId_t OWNERSHIP_QOS_POLICY_ID` = `QosPolicyId_t.OWNERSHIP_QOS_POLICY_ID`
- const `QosPolicyId_t OWNERSHIPSTRENGTH_QOS_POLICY_ID` = `QosPolicyId_t.OWNERSHIPSTRENGTH_QOS_POLICY_ID`
- const `QosPolicyId_t LIVELINESS_QOS_POLICY_ID` = `QosPolicyId_t.LIVELINESS_QOS_POLICY_ID`
- const `QosPolicyId_t TIMEBASEDFILTER_QOS_POLICY_ID` = `QosPolicyId_t.TIMEBASEDFILTER_QOS_POLICY_ID`
- const `QosPolicyId_t PARTITION_QOS_POLICY_ID` = `QosPolicyId_t.PARTITION_QOS_POLICY_ID`
- const `QosPolicyId_t RELIABILITY_QOS_POLICY_ID` = `QosPolicyId_t.RELIABILITY_QOS_POLICY_ID`
- const `QosPolicyId_t DESTINATIONORDER_QOS_POLICY_ID` = `QosPolicyId_t.DESTINATIONORDER_QOS_POLICY_ID`
- const `QosPolicyId_t HISTORY_QOS_POLICY_ID` = `QosPolicyId_t.HISTORY_QOS_POLICY_ID`
- const `QosPolicyId_t RESOURCELIMITS_QOS_POLICY_ID` = `QosPolicyId_t.RESOURCELIMITS_QOS_POLICY_ID`
- const `QosPolicyId_t ENTITYFACTORY_QOS_POLICY_ID` = `QosPolicyId_t.ENTITYFACTORY_QOS_POLICY_ID`
- const `QosPolicyId_t WRITERDATALIFECYCLE_QOS_POLICY_ID` = `QosPolicyId_t.WRITERDATALIFECYCLE_QOS_POLICY_ID`
- const `QosPolicyId_t READERDATALIFECYCLE_QOS_POLICY_ID` = `QosPolicyId_t.READERDATALIFECYCLE_QOS_POLICY_ID`
- const `QosPolicyId_t TOPICDATA_QOS_POLICY_ID` = `QosPolicyId_t.TOPICDATA_QOS_POLICY_ID`
- const `QosPolicyId_t GROUPDATA_QOS_POLICY_ID` = `QosPolicyId_t.GROUPDATA_QOS_POLICY_ID`
- const `QosPolicyId_t TRANSPORTPRIORITY_QOS_POLICY_ID` = `QosPolicyId_t.TRANSPORTPRIORITY_QOS_POLICY_ID`
- const `QosPolicyId_t LIFESPAN_QOS_POLICY_ID` = `QosPolicyId_t.LIFESPAN_QOS_POLICY_ID`
- const `QosPolicyId_t DURABILITYSERVICE_QOS_POLICY_ID` = `QosPolicyId_t.DURABILITYSERVICE_QOS_POLICY_ID`
- const `ReturnCode_t RETCODE_OK` = `ReturnCode_t.RETCODE_OK`
- const `ReturnCode_t RETCODE_ERROR` = `ReturnCode_t.RETCODE_ERROR`
- const `ReturnCode_t RETCODE_UNSUPPORTED` = `ReturnCode_t.RETCODE_UNSUPPORTED`
- const `ReturnCode_t RETCODE_BAD_PARAMETER` = `ReturnCode_t.RETCODE_BAD_PARAMETER`

- const `ReturnCode_t RETCODE_PRECONDITION_NOT_MET` = `ReturnCode_t.RETCODE_PRECONDITION_NOT_MET`
- const `ReturnCode_t RETCODE_OUT_OF_RESOURCES` = `ReturnCode_t.RETCODE_OUT_OF_RESOURCES`
- const `ReturnCode_t RETCODE_NOT_ENABLED` = `ReturnCode_t.RETCODE_NOT_ENABLED`
- const `ReturnCode_t RETCODE_IMMUTABLE_POLICY` = `ReturnCode_t.RETCODE_IMMUTABLE_POLICY`
- const `ReturnCode_t RETCODE_INCONSISTENT_POLICY` = `ReturnCode_t.RETCODE_INCONSISTENT_POLICY`
- const `ReturnCode_t RETCODE_ALREADY_DELETED` = `ReturnCode_t.RETCODE_ALREADY_DELETED`
- const `ReturnCode_t RETCODE_TIMEOUT` = `ReturnCode_t.RETCODE_TIMEOUT`
- const `ReturnCode_t RETCODE_NO_DATA` = `ReturnCode_t.RETCODE_NO_DATA`
- const `DomainParticipantQos PARTICIPANT_QOS_DEFAULT` = `null`
- const `TopicQos TOPIC_QOS_DEFAULT` = `null`
- const `PublisherQos PUBLISHER_QOS_DEFAULT` = `null`
- const `SubscriberQos SUBSCRIBER_QOS_DEFAULT` = `null`
- const `DataWriterQos DATAWRITER_QOS_DEFAULT` = `null`
- const `DataReaderQos DATAREADER_QOS_DEFAULT` = `null`

### Static Public Attributes

- static readonly `InstanceHandle_t HANDLE_NIL` = `new InstanceHandle_t((IntPtr)0)`

### 3.9.1 Detailed Description

The 'DDS' class includes several convenient constants.





### 3.9.2 Member Data Documentation

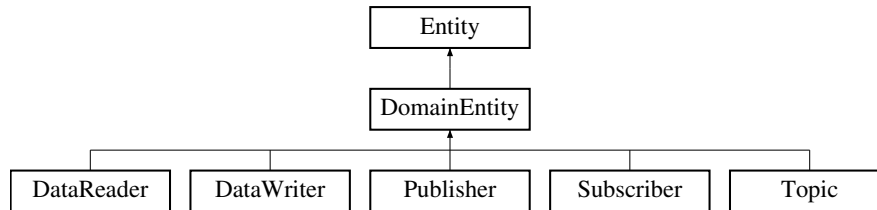
- 3.9.2.1 `const uint ALIVE_INSTANCE_STATE = 0x0001`
- 3.9.2.2 `const uint ALL_STATUS = 0xffff`
- 3.9.2.3 `const uint ANY_INSTANCE_STATE = 0x00FF`
- 3.9.2.4 `const uint ANY_SAMPLE_STATE = 0x00FF`
- 3.9.2.5 `const uint ANY_VIEW_STATE = 0x00FF`
- 3.9.2.6 `const uint DATA_AVAILABLE_STATUS = 0x0400`
- 3.9.2.7 `const uint DATA_ON_READERS_STATUS = 0x0200`
- 3.9.2.8 `const DataReaderQos DATAREADER_QOS_DEFAULT = null`
- 3.9.2.9 `const DataWriterQos DATAWRITER_QOS_DEFAULT = null`
- 3.9.2.10 `const QosPolicyId_t DEADLINE_QOS_POLICY_ID = QosPolicyId_t.DEADLINE_QOS_POLICY_ID`
- 3.9.2.11 `const QosPolicyId_t DESTINATIONORDER_QOS_POLICY_ID = QosPolicyId_t.DESTINATIONORDER_QOS_POLICY_ID`
- 3.9.2.12 `const QosPolicyId_t DURABILITY_QOS_POLICY_ID = QosPolicyId_t.DURABILITY_QOS_POLICY_ID`
- 3.9.2.13 `const QosPolicyId_t DURABILITYSERVICE_QOS_POLICY_ID = QosPolicyId_t.DURABILITYSERVICE_QOS_POLICY_ID`
- 3.9.2.14 `const uint DURATION_INFINITE_NSEC = 0xffffffff`
- 3.9.2.15 `const int DURATION_INFINITE_SEC = 0x7fffffff`
- 3.9.2.16 `const int DURATION_ZERO_NSEC = 0`
- 3.9.2.17 `const int DURATION_ZERO_SEC = 0`
- 3.9.2.18 `const QosPolicyId_t ENTITYFACTORY_QOS_POLICY_ID = QosPolicyId_t.ENTITYFACTORY_QOS_POLICY_ID`
- 3.9.2.19 `const QosPolicyId_t GROUPEXAMPLE_QOS_POLICY_ID = QosPolicyId_t.GROUPEXAMPLE_QOS_POLICY_ID`
- 3.9.2.20 `readonly InstanceHandle_t HANDLE_NIL = new InstanceHandle_t((IntPtr)0) [static]`  
Generated on Sun Jan 1 2012 10:24:56 for CoreDDS Data Distribution Service by Doxygen
- 3.9.2.21 `const QosPolicyId_t HISTORY_QOS_POLICY_ID = QosPolicyId_t.HISTORY_QOS_POLICY_ID`
- 3.9.2.22 `const uint INCONSISTENT_TOPIC_STATUS = 0x0001`
- 3.9.2.23 `const QosPolicyId_t LATENCYBUDGET_QOS_POLICY_ID = QosPolicyId_t.LATENCYBUDGET_QOS_POLICY_ID`

- `/home/ctucker/coredx_v3.4rc/src/dds_csharp/base.cs`

## 3.10 DomainEntity Class Reference

Base class for all DDS Domain Entities.

Inheritance diagram for DomainEntity:



### 3.10.1 Detailed Description

Base class for all DDS Domain Entities.

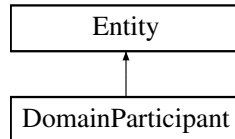
The documentation for this class was generated from the following file:

- `/home/ctucker/coredx_v3.4rc/src/dds_csharp/entity.cs`

### 3.11 DomainParticipant Class Reference

The [DomainParticipant](#) is used to configure, create and destroy [Publisher](#), [Subscriber](#) and [Topic](#) objects.

Inheritance diagram for DomainParticipant:



#### Public Member Functions

- override [ReturnCode\\_t enable](#) ()  
*Enables the [DomainParticipant](#).*
- override [InstanceHandle\\_t get\\_instance\\_handle](#) ()
- [ReturnCode\\_t get\\_qos](#) ([DomainParticipantQos](#) qos)
- [ReturnCode\\_t set\\_qos](#) ([DomainParticipantQos](#) qos)
- [ReturnCode\\_t set\\_listener](#) ([DomainParticipantListener](#) new\_listener, uint mask)
- [DomainParticipantListener get\\_listener](#) ()
- [Publisher create\\_publisher](#) ([PublisherQos](#) qos, [PublisherListener](#) listener, uint mask)
- [ReturnCode\\_t delete\\_publisher](#) ([Publisher](#) p)
- [Subscriber create\\_subscriber](#) ([SubscriberQos](#) qos, [SubscriberListener](#) listener, uint mask)
- [ReturnCode\\_t delete\\_subscriber](#) ([Subscriber](#) s)
- [Topic create\\_topic](#) (string topic\_name, string type\_name, [TopicQos](#) qos, [TopicListener](#) listener, uint mask)
- [ReturnCode\\_t delete\\_topic](#) ([Topic](#) topic)
- [ContentFilteredTopic create\\_contentfilteredtopic](#) (String name, [Topic](#) related\_topic, String filter\_expression, List< String > filter\_parameters)
- [ReturnCode\\_t delete\\_contentfilteredtopic](#) ([ContentFilteredTopic](#) cft)
- [MultiTopic create\\_multitopic](#) (String name, String type\_name, String subscription\_expression, List< String > expression\_params)
- [ReturnCode\\_t delete\\_multitopic](#) ([MultiTopic](#) a\_multitopic)
- [Topic find\\_topic](#) (String topic\_name, [Duration\\_t](#) timeout)
- [TopicDescription lookup\\_topicdescription](#) (String name)
- [Subscriber get\\_built\\_in\\_subscriber](#) ()
- [ReturnCode\\_t ignore\\_participant](#) ([InstanceHandle\\_t](#) handle)
- [ReturnCode\\_t ignore\\_topic](#) ([InstanceHandle\\_t](#) handle)
- [ReturnCode\\_t ignore\\_publication](#) ([InstanceHandle\\_t](#) handle)
- [ReturnCode\\_t ignore\\_subscription](#) ([InstanceHandle\\_t](#) handle)

- long `get_domain_id ()`
- `ReturnCode_t delete_contained_entities ()`
- `ReturnCode_t assert_liveliness ()`
- `ReturnCode_t set_default_publisher_qos (PublisherQos qos)`
- `ReturnCode_t get_default_publisher_qos (PublisherQos qos)`
- `ReturnCode_t set_default_subscriber_qos (SubscriberQos qos)`
- `ReturnCode_t get_default_subscriber_qos (SubscriberQos qos)`
- `ReturnCode_t set_default_topic_qos (TopicQos qos)`
- `ReturnCode_t get_default_topic_qos (TopicQos qos)`
- `ReturnCode_t get_discovered_participants (List< InstanceHandle_t > participant_handles)`
- `ReturnCode_t get_discovered_participant_data (ParticipantBuiltinTopicData participant_data, InstanceHandle_t participant_handle)`
- `ReturnCode_t get_discovered_topics (List< InstanceHandle_t > topic_handles)`
- `ReturnCode_t get_discovered_topic_data (TopicBuiltinTopicData topic_data, InstanceHandle_t topic_handle)`
- bool `contains_entity (InstanceHandle_t handle)`
- `ReturnCode_t get_current_time (Time_t current_time)`

### 3.11.1 Detailed Description

The `DomainParticipant` is used to configure, create and destroy `Publisher`, `Subscriber` and `Topic` objects. The `DomainParticipant` is a container for the objects that it creates. The objects operate within the **domain** identified by the `domain_id` provided when the `DomainParticipant` was created. Objects within different **domains** do not communicate or interfere with each other.

### 3.11.2 Member Function Documentation

#### 3.11.2.1 `ReturnCode_t assert_liveliness ( ) [inline]`

This operation manually asserts the liveliness of the `DomainParticipant dp`. This operation indicates that the `DomainParticipant` is still alive. It is effective only if the `DomainParticipant` contains one or more `DataWriter` objects with LIVENESS QoS set to `MANUAL_BY_PARTICIPANT_LIVENESS_QOS`. In this case, the operation will assert the liveliness of those particular `DataWriter` objects.

Writing data via the `DataWriter write` operation automatically asserts the liveliness of the `DataWriter` and its containing `DomainParticipant`. Therefore, the use of `DomainParticipant.assert_liveliness()` is needed only if the application is not writing data frequently enough to keep the `DataWriter` considered 'alive'.

#### 3.11.2.2 `bool contains_entity ( InstanceHandle_t handle ) [inline]`

This operation checks whether or not the given handle `a_handle` represents an object that was created by the `DomainParticipant d`. This applies recursively to all objects created by the `DomainParticipant`.

The routine will return true (non-zero) if the handle is found, zero otherwise.

### 3.11.2.3 ContentFilteredTopic `create_contentfilteredtopic ( String name, Topic related_topic, String filter_expression, List< String > filter_parameters ) [inline]`

This operation creates a [ContentFilteredTopic](#).

The [ContentFilteredTopic](#) is associated with another un-filtered topic **related\_topic**.

The **filter\_expression** is an SQL like condition expression, and **filter\_parameters** provide optional parameters that are referenced by the **filter\_expression**. The syntax of the filter expression is similar to the WHERE clause in SQL. For example "x<4" is a valid filter expression. It would test that data member 'x' is less than value '4'. If the filter expression evaluates to TRUE, then the data sample will be available, otherwise the data sample would be 'filtered' (excluded).

The filter\_expression can refer to parameters. The syntax for parameters is the percent sign "%" followed by a number. The number is the index of the parameter in the **filter\_parameters** sequence. Parameters are counted starting at zero. So, "%0" refers to the first parameter, and "%4" refers to the fifth parameter. Using this syntax, the expression "x<%0" would test the value of 'x' against the first parameter in the sequence.

### 3.11.2.4 MultiTopic `create_multitopic ( String name, String type_name, String subscription_expression, List< String > expression_params ) [inline]`

#### Not Yet Supported

This is currently unsupported in the C# language binding.

### 3.11.2.5 Publisher `create_publisher ( PublisherQos qos, PublisherListener listener, uint mask ) [inline]`

This operation creates a [Publisher](#) with the specified [PublisherQoS](#) settings and [PublisherListener](#). The value **DDS.PUBLISHER\_QOS\_DEFAULT** may be provided for the **qos** argument. This will indicate that the default publisher QoS settings held in the [DomainParticipant](#) should be used.

This operation may fail and return NULL if the QoS settings are internally inconsistent.

### 3.11.2.6 Subscriber `create_subscriber ( SubscriberQos qos, SubscriberListener listener, uint mask ) [inline]`

This operation creates a [Subscriber](#) with the specified [SubscriberQoS](#) and [SubscriberListener](#). The value **DDS.SUBSCRIBER\_QOS\_DEFAULT** may be provided for the **qos** argument. This will indicate that the default subscriber QoS settings held in the [DomainParticipant](#) should be used.

This operation may fail and return NULL if the QoS settings are internally inconsistent.

### 3.11.2.7 Topic create\_topic ( string topic\_name, string type\_name, TopicQos qos, TopicListener listener, uint mask ) [inline]

This operation creates a [Topic](#) with the specified [TopicQos](#) settings and [TopicListener](#).

The value [DDS.TOPIC\\_QOS\\_DEFAULT](#) may be provided for the **qos** argument. This will indicate that the the default topic QoS settings held in the [DomainParticipant](#) should be used.

The topic is created with a reference to the data type indicated by the **type\_name** argument. The data type must have been registered with the [DomainParticipant](#) (by a call to [register\\_type\(\)](#) on the appropriate [TypeSupport](#) object) prior to calling [create\\_topic\(\)](#).

This operation may fail and return NULL if the QoS settings are internally inconsistent.

The topic returned from this operation (if not NULL) must be deleted by calling [DomainParticipant.delete\\_topic\(\)](#).

### 3.11.2.8 ReturnCode\_t delete\_contained\_entities ( ) [inline]

This operation deletes all the objects created by means of the **create** operations on [DomainParticipant dp](#). This routine will recursively call the corresponding [delete\\_contained\\_entities\(\)](#) operation on each of the contained objects (Publishers, Subscribers, Topics, ContentFilteredTopics, and MultiTopics). If successful, this operation will recursively delete all objects contained with this [DomainParticipant](#). After successful execution, the application may delete the [DomainParticipant](#) by calling [DomainParticipantFactory.delete\\_participant\(\)](#).

If any of the objects cannot be deleted, this routine will return [ReturnCode\\_t.RETCODE\\_PRECONDITION\\_NOT\\_MET](#).

### 3.11.2.9 ReturnCode\_t delete\_contentfilteredtopic ( ContentFilteredTopic cft ) [inline]

This operation deletes a previously allocated [ContentFilteredTopic](#). This operation will fail if there are any [DataReader](#), [DataWriter](#), [ContentFilteredTopic](#), or [MultiTopic](#) objects that reference the specified [Topic](#). In this case, the operation will return [ReturnCode\\_t.RETCODE\\_PRECONDITION\\_NOT\\_MET](#).

The [Topic](#) must be owned by [DomainParticipant dp](#); otherwise [ReturnCode\\_t.RETCODE\\_PRECONDITION\\_NOT\\_MET](#) will be returned.

### 3.11.2.10 ReturnCode\_t delete\_multitopic ( MultiTopic a\_multitopic ) [inline]

#### Not Yet Supported

This is currently unsupported in the C# language binding.

### 3.11.2.11 `ReturnCode_t delete_publisher ( Publisher p ) [inline]`

This operation deletes an existing [Publisher](#). A [Publisher](#) cannot be deleted if it contains any [DataWriter](#) objects. In this case, `delete_publisher` will return `ReturnCode_t.RETCODE_PRECONDITION_NOT_MET`.

If the [DomainParticipant](#) `dp` does not contain the provided [Publisher](#) `p`, the operation will return `ReturnCode_t.RETCODE_PRECONDITION_NOT_MET`.

### 3.11.2.12 `ReturnCode_t delete_subscriber ( Subscriber s ) [inline]`

This operation deletes an existing [Subscriber](#). A [Subscriber](#) cannot be deleted if it contains any [DataReader](#) objects. In this case, `delete_subscriber` will return `ReturnCode_t.RETCODE_PRECONDITION_NOT_MET`.

If the [DomainParticipant](#) `dp` does not contain the provided [Subscriber](#) `s`, the operation will return `ReturnCode_t.RETCODE_PRECONDITION_NOT_MET`.

### 3.11.2.13 `ReturnCode_t delete_topic ( Topic topic ) [inline]`

This operation deletes a [Topic](#). This operation will fail if there are any [DataReader](#), [DataWriter](#), [ContentFilteredTopic](#), or [MultiTopic](#) objects that reference the specified [Topic](#). In this case, the operation will return `ReturnCode_t.RETCODE_PRECONDITION_NOT_MET`.

The [Topic](#) must be owned by [DomainParticipant](#) `dp`; otherwise `ReturnCode_t.RETCODE_PRECONDITION_NOT_MET` will be returned.

### 3.11.2.14 `override ReturnCode_t enable ( ) [inline, virtual]`

Enables the [DomainParticipant](#).

This is crappy

A [DomainParticipant](#) is created either enabled or not based on the `DomainParticipantFactoryQoS` setting `entity_factory`. When a [DomainParticipant](#) is not enabled, only the following sub-set of all [DomainParticipant](#) operations are legal:

- operations to get and set QoS policies,
- factory operations (create, delete),
- `get_statuscondition()`,
- `get_status_changes()`,
- lookup operations

Any other [DomainParticipant](#) operation will return the `ReturnCode_t.RETCODE_NOT_ENABLED` error. `DomainParticipant.enable()` may be called on an already enabled [DomainParticipant](#) [it will have no effect].



Reimplemented from [Entity](#).

### 3.11.2.15 Topic `find_topic ( String topic_name, Duration_t timeout ) [inline]`

This operation returns a [Topic](#) identified by the provided **topic\_name**. If a topic with name **topic\_name** is known to exist, the function returns this matching topic immediately. Otherwise, the operation will block for up to the **timeout** duration. If a topic with name **topic\_name** is discovered or otherwise created, the operation will cease to wait, and will return the matching topic.

If a matching topic is not known by the time the **timeout** has expired, the operation will return NULL.

Like `DomainParticipant_create_topic()`, the topic returned from this operation (if not NULL) must be deleted by calling `DomainParticipant.delete_topic()`.

### 3.11.2.16 Subscriber `get_builtin_subscriber ( ) [inline]`

Returns the built-in [Subscriber](#).

Each [DomainParticipant](#) contains several built-in [Topic](#) objects as well as corresponding [DataReader](#) objects to access them. These built-in [DataReader](#) objects belong to the single built-in [Subscriber](#) that is accessed through this operation.

The built-in Topics are used to communicate information about other [DomainParticipant](#), [Topic](#), [DataReader](#), and [DataWriter](#) objects.

An application should not explicitly delete the [Subscriber](#) returned by this operation - it is managed internally.

#### See also

- ParticipantBuiltinTopicDataDataReader
- PublicationBuiltinTopicDataDataReader
- SubscriptionBuiltinTopicDataDataReader

### 3.11.2.17 `ReturnCode_t get_current_time ( Time_t current_time ) [inline]`

This operation returns the current value of the time used by the service.

### 3.11.2.18 `ReturnCode_t get_default_publisher_qos ( PublisherQos qos ) [inline]`

Provides access to the default [PublisherQos](#) settings held in the factory. The provided **qos** argument is populated with the default qos settings.

### 3.11.2.19 `ReturnCode_t get_default_subscriber_qos ( SubscriberQos qos ) [inline]`

Provides access to the default [SubscriberQos](#) settings held in the factory. The provided `qos` argument is populated with the default qos settings.

### 3.11.2.20 `ReturnCode_t get_default_topic_qos ( TopicQos qos ) [inline]`

Provides access to the default [TopicQos](#) settings held in the factory. The provided `qos` argument is populated with the default qos settings.

### 3.11.2.21 `ReturnCode_t get_discovered_participant_data ( ParticipantBuiltinTopicData participant_data, InstanceHandle_t participant_handle ) [inline]`

This operation returns data that describes a particular discovered participant identified by `participant_handle`. An appropriate handle can be obtained through a call to [DomainParticipant.get\\_discovered\\_participants\(\)](#).

If `participant_handle` does not identify a known [DomainParticipant](#), this routine will return `ReturnCode_t.RETCODE_PRECONDITION_NOT_MET`.

### 3.11.2.22 `ReturnCode_t get_discovered_participants ( List< InstanceHandle_t > participant_handles ) [inline]`

This operation returns the list of handles identifying the [DomainParticipant](#) objects that have been discovered. The returned list will include only participants which are in the same domain as participant `d`, and which are not explicitly ignored as a result of a call to [DomainParticipant.ignore\\_participant\(\)](#).

### 3.11.2.23 `ReturnCode_t get_discovered_topic_data ( TopicBuiltinTopicData topic_data, InstanceHandle_t topic_handle ) [inline]`

This operation returns data that describes a particular discovered topic identified by `topic_handle`. An appropriate handle can be obtained through a call to [DomainParticipant.get\\_discovered\\_topics\(\)](#).

If `topic_handle` does not identify a known [Topic](#), this routine will return `ReturnCode_t.RETCODE_PRECONDITION_NOT_MET`.

## Not Yet Supported

This is currently unsupported in the C# language binding.

**3.11.2.24 ReturnCode\_t get\_discovered\_topics ( List< InstanceHandle\_t > *topic\_handles* ) [inline]**

This operation returns the list of handles identifying the [Topic](#) objects that have been discovered. The returned list will include only topics which are in the same domain as participant **d**, and which are not explicitly ignored as a result of a call to [DomainParticipant.ignore\\_topic\(\)](#).

**Not Yet Supported**

This is currently unsupported in the C# language binding.

**3.11.2.25 long get\_domain\_id ( ) [inline]**

Gets the **domain\_id** to which the [DomainParticipant](#) belongs.

**3.11.2.26 override InstanceHandle\_t get\_instance\_handle ( ) [inline, virtual]**

Gets the handle that locally identifies this [Entity](#).

Reimplemented from [Entity](#).

**3.11.2.27 DomainParticipantListener get\_listener ( ) [inline]**

This operation returns the currently installed [DomainParticipantListener](#). Because the infrastructure makes a copy of the listener provided in [DomainParticipant.set\\_listener\(\)](#), the returned structure pointer will not match the pointer originally provided. However, the function pointers within the structure will match. Also, the application should not free the data referenced by the returned pointer.

**3.11.2.28 ReturnCode\_t get\_qos ( DomainParticipantQos *qos* ) [inline]**

Gets the [DomainParticipantQoS](#) settings of the [DomainParticipant](#).

**3.11.2.29 ReturnCode\_t ignore\_participant ( InstanceHandle\_t *handle* ) [inline]**

Instructs the [DomainParticipant dp](#) to ignore the external [DomainParticipant](#) identified by **handle**. This will cause the infrastructure to behave as if the external participant **handle** did not exist. The handle can be discovered by accessing the built-in topic [DCPSParticipant](#) via the appropriate built-in [DataReader](#).

There is no mechanism to reverse this operation.

**Not Yet Supported**

This is currently unsupported in the C# language binding.

### 3.11.2.30 `ReturnCode_t ignore_publication ( InstanceHandle_t handle ) [inline]`

This operation instructs the [DomainParticipant dp](#) to ignore a Publication identified by **handle**. The handle can be discovered by accessing the built-in topic **DCPSPublication** via the appropriate built-in [DataReader](#).

There is no mechanism to reverse this operation.

#### Not Yet Supported

This is currently unsupported in the C# language binding.

### 3.11.2.31 `ReturnCode_t ignore_subscription ( InstanceHandle_t handle ) [inline]`

This operation instructs the [DomainParticipant dp](#) to ignore a Subscription identified by **handle**. The handle can be discovered by accessing the built-in topic **DCPSSubscription** via the appropriate built-in [DataReader](#).

There is no mechanism to reverse this operation.

#### Not Yet Supported

This is currently unsupported in the C# language binding.

### 3.11.2.32 `ReturnCode_t ignore_topic ( InstanceHandle_t handle ) [inline]`

This operation instructs the [DomainParticipant dp](#) to ignore a [Topic](#) identified by **handle**. This can be used to save resources if a participant will never participate on certain Topics. The handle can be discovered by accessing the built-in topic **DCPSTopic** via the appropriate built-in [DataReader](#).

There is no mechanism to reverse this operation.

#### Not Yet Supported

This is currently unsupported in the C# language binding.

### 3.11.2.33 `TopicDescription lookup_topicdescription ( String name ) [inline]`

This operation returns an existing, locally-created [TopicDescription](#), named **name**. If a [TopicDescription](#) named **name** does not exist, then this routine will return NULL.

### 3.11.2.34 `ReturnCode_t set_default_publisher_qos ( PublisherQos qos ) [inline]`

Sets the default [PublisherQos](#) held in the [DomainParticipant](#). This default qos will be used during subsequent calls to [DomainParticipant.create\\_publisher\(\)](#) if the special **DDS.PUBLISHER\_QOS\_DEFAULT** value is provided for **qos**. This routine may fail if the provided **qos** argument is not internally consistent. In this

case, `ReturnCode_t.RETCODE_INCONSISTENT_POLICY` will be returned, and no changes will be made to the [DomainParticipant](#).

#### 3.11.2.35 `ReturnCode_t set_default_subscriber_qos ( SubscriberQos qos ) [inline]`

Sets the default [SubscriberQos](#) held in the [DomainParticipant](#). This default qos will be used during subsequent calls to [DomainParticipant.create\\_subscriber\(\)](#) if the special `DDS.SUBSCRIBER_QOS_DEFAULT` value is provided for `qos`. This routine may fail if the provided `qos` argument is not internally consistent. In this case, `ReturnCode_t.RETCODE_INCONSISTENT_POLICY` will be returned, and no changes will be made to the [DomainParticipant](#).

#### 3.11.2.36 `ReturnCode_t set_default_topic_qos ( TopicQos qos ) [inline]`

Sets the default [TopicQos](#) held in the [DomainParticipant](#). This default qos will be used during subsequent calls to [DomainParticipant.create\\_topic\(\)](#) [and related] if the special `DDS.TOPIC_QOS_DEFAULT` value is provided for `qos`. This routine may fail if the provided `qos` argument is not internally consistent. In this case, `ReturnCode_t.RETCODE_INCONSISTENT_POLICY` will be returned, and no changes will be made to the [DomainParticipant](#).

#### 3.11.2.37 `ReturnCode_t set_listener ( DomainParticipantListener new_listener, uint mask ) [inline]`

This operation installs a [DomainParticipantListener](#) on the [DomainParticipant](#). Only one listener may be attached to a [DomainParticipant](#) at a time. A call to [set\\_listener\(\)](#) will replace any current listener with `a_listener`.

`a_listener` can be `NULL`, which indicates a listener that does nothing.

The infrastructure will make an internal copy of the listener structure so that it need not be persisted by the application.

#### 3.11.2.38 `ReturnCode_t set_qos ( DomainParticipantQos qos ) [inline]`

Sets the [DomainParticipantQoS](#) values. These QoS values affect the behavior of the [DomainParticipant](#). This routine may fail if the provided `qos` argument is not internally consistent. In this case, `ReturnCode_t.RETCODE_INCONSISTENT_POLICY` will be returned, and no changes will be made to the [DomainParticipant](#) QoS.

The documentation for this class was generated from the following file:

- `/home/ctucker/coredx_v3.4rc/src/dds_csharp/dp.cs`

## 3.12 DomainParticipantFactory Class Reference

[DomainParticipantFactory](#) constructs [DomainParticipants](#). The.

### Public Member Functions

- [DomainParticipant](#) [create\\_participant](#) (uint domain\_id, [DomainParticipantQos](#) qos, [DomainParticipantListener](#) listener, uint mask)

*This operation creates a new [DomainParticipant](#) object. The caller provides the domain\_id to which the Participant should belong. The listener and mask arguments are used to specify a set of callback routines which will be invoked upon detection of certain events. The qos argument specifies the [DomainParticipant](#) Quality of Service settings that should be used when creating the [DomainParticipant](#). It may be specified as [DDS\\_PARTICIPANT\\_QOS\\_DEFAULT](#) to instruct [CoreDX](#) to use the default qos settings held in the [DomainParticipantFactory](#). This routine will return NULL if it fails to create a [DomainParticipant](#).*

### Static Public Member Functions

- static [DomainParticipantFactory](#) [get\\_instance](#) ()

### Properties

- static [DomainParticipantFactory](#) [Instance](#) [get]

### Related Functions

(Note that these are not member functions.)

- [ReturnCode\\_t](#) [delete\\_participant](#) ([DomainParticipant](#) participant)  
*Destroys the provided [DomainParticipant](#).*
- [DomainParticipant](#) [lookup\\_participant](#) (uint domain\_id)  
*Returns a previously created [DomainParticipant](#) belonging to the specified **domain\_id**.*
- [ReturnCode\\_t](#) [set\\_default\\_participant\\_qos](#) ([DomainParticipantQos](#) qos)  
*Sets the default [DDS\\_DomainParticipantQos](#) held in the factory.*
- [ReturnCode\\_t](#) [get\\_default\\_participant\\_qos](#) ([DomainParticipantQos](#) qos)  
*Provides access to the default [Participant qos](#) held in the factory.*
- [ReturnCode\\_t](#) [set\\_qos](#) ([DomainParticipantFactoryQos](#) qos)

Sets the *DomainParticipantFactory* QoS values.

- `ReturnCode_t get_qos (DomainParticipantFactoryQos qos)`

Provides access to the QoS settings of the *DomainParticipantFactory*.

### 3.12.1 Detailed Description

`DomainParticipantFactory` constructs `DomainParticipant`s. The `DomainParticipantFactory` is used to configure, create and destroy `DomainParticipant` instances.

#### Author

Twin Oaks Computing, Inc

### 3.12.2 Member Function Documentation

#### 3.12.2.1 `DomainParticipant create_participant ( uint domain_id, DomainParticipantQos qos, DomainParticipantListener listener, uint mask ) [inline]`

This operation creates a new `DomainParticipant` object. The caller provides the `domain_id` to which the Participant should belong. The listener and mask arguments are used to specify a set of callback routines which will be invoked upon detection of certain events. The qos argument specifies the `DomainParticipant` Quality of Service settings that should be used when creating the `DomainParticipant`. It may be specified as `DDS_PARTICIPANT_QOS_DEFAULT` to instruct CoreDX to use the default qos settings held in the `DomainParticipantFactory`. This routine will return NULL if it fails to create a `DomainParticipant`.

#### Returns

`DomainParticipant`

#### 3.12.2.2 `static DomainParticipantFactory get_instance ( ) [inline, static]`

Get access to the singleton `DomainParticipantFactory`.

### 3.12.3 Friends And Related Function Documentation

#### 3.12.3.1 `ReturnCode_t delete_participant ( DomainParticipant participant ) [related]`

Destroys the provided `DomainParticipant`.

This routine will fail if all Entities (Publishers, Subscribers, etc) created through the specified `DomainParticipant` have not yet been deleted. (In this case, `DDS_RETCODE_PRECONDITION_NOT_MET` will be returned.)

### 3.12.3.2 `ReturnCode_t get_default_participant_qos ( DomainParticipantQos qos ) [related]`

Provides access to the default Participant qos held in the factory.

The provided `qos` argument is populated with the default qos settings.

### 3.12.3.3 `DomainParticipant lookup_participant ( uint domain_id ) [related]`

Returns a previously created [DomainParticipant](#) belonging to the specified `domain_id`.

If there are multiple DomainParticipants in existence within the specified domain, one of them will be returned.

### 3.12.3.4 `ReturnCode_t set_default_participant_qos ( DomainParticipantQos qos ) [related]`

Sets the default DDS\_DomainParticipantQos held in the factory.

This default qos will be used during subsequent calls to `DDS_DomainParticipantFactory_create_participant()` if the special `DDS_PARTICIPANT_QOS_DEFAULT` value is provided for `qos`.

This routine may fail if the provided `qos` argument is not internally consistent. In this case, `DDS_INCONSISTENT_POLICY` will be returned, and no changes will be made to the [DomainParticipantFactory](#).

### 3.12.3.5 `ReturnCode_t set_qos ( DomainParticipantFactoryQos qos ) [related]`

Sets the [DomainParticipantFactory](#) QoS values.

These QoS values affect the behavior of the factory.

This routine may fail if the provided `qos` argument is not internally consistent. In this case, `DDS_INCONSISTENT_POLICY` will be returned, and no changes will be made to the [DomainParticipantFactory](#).

## 3.12.4 Property Documentation

### 3.12.4.1 `DomainParticipantFactory Instance [static, get]`

A C# style property with accessor. use it like this: `DomainParticipantFactory.Instance.create_participant()`

The documentation for this class was generated from the following file:

- `/home/ctucker/coredx_v3.4rc/src/dds_csharp/dpf.cs`



## 3.13 DomainParticipantFactoryQos Class Reference

Structure that holds [DomainParticipantFactory](#) Quality of Service policies.

### Public Attributes

- [EntityFactoryQosPolicy](#) entity\_factory

### 3.13.1 Detailed Description

Structure that holds [DomainParticipantFactory](#) Quality of Service policies.

#### See also

- [DomainParticipantFactory::set\\_qos\(\)](#)
- [DomainParticipantFactory::get\\_qos\(\)](#)

### 3.13.2 Member Data Documentation

#### 3.13.2.1 EntityFactoryQosPolicy entity\_factory

Controls the behavior of the [DomainParticipant](#) 'create' operations.

The documentation for this class was generated from the following file:

- /home/ctucker/coredx\_v3.4rc/src/dds\_csharp/base.cs

## 3.14 DomainParticipantListener Class Reference

The [DomainParticipantListener](#) provides asynchronous notification of [DomainParticipant](#) events.

### Public Attributes

- `inconsistent_topic_delegate` [on\\_inconsistent\\_topic](#)
- `offered_deadline_missed_delegate` [on\\_offered\\_deadline\\_missed](#)
- `offered_incompatible_qos_delegate` [on\\_offered\\_incompatible\\_qos](#)
- `liveliness_lost_delegate` [on\\_liveliness\\_lost](#)
- `publication_matched_delegate` [on\\_publication\\_matched](#)
- `requested_deadline_missed_delegate` [on\\_requested\\_deadline\\_missed](#)
- `requested_incompatible_qos_delegate` [on\\_requested\\_incompatible\\_qos](#)
- `sample_rejected_delegate` [on\\_sample\\_rejected](#)
- `liveliness_changed_delegate` [on\\_liveliness\\_changed](#)
- `data_available_delegate` [on\\_data\\_available](#)
- `subscription_matched_delegate` [on\\_subscription\\_matched](#)
- `sample_lost_delegate` [on\\_sample\\_lost](#)
- `data_on_readers_delegate` [on\\_data\\_on\\_readers](#)

### 3.14.1 Detailed Description

The [DomainParticipantListener](#) provides asynchronous notification of [DomainParticipant](#) events. This listener can be installed during [DomainParticipant](#) creation, [DomainParticipantFactory.create\\_participant\(\)](#), as well as by calling [DomainParticipantset\\_listener\(\)](#).

#### Note

The listener callback methods should be lightweight and should not block. If a callback method blocks, it will block all other callback operations within the same [DomainParticipant](#).

## 3.14.2 Member Data Documentation

3.14.2.1 `data_available_delegate` on `data_available`

3.14.2.2 `data_on_readers_delegate` on `data_on_readers`

3.14.2.3 `inconsistent_topic_delegate` on `inconsistent_topic`

3.14.2.4 `liveliness_changed_delegate` on `liveliness_changed`

3.14.2.5 `liveliness_lost_delegate` on `liveliness_lost`

3.14.2.6 `offered_deadline_missed_delegate` on `offered_deadline_missed`

3.14.2.7 `offered_incompatible_qos_delegate` on `offered_incompatible_qos`

3.14.2.8 `publication_matched_delegate` on `publication_matched`

3.14.2.9 `requested_deadline_missed_delegate` on `requested_deadline_missed`

3.14.2.10 `requested_incompatible_qos_delegate` on `requested_incompatible_qos`

3.14.2.11 `sample_lost_delegate` on `sample_lost`

3.14.2.12 `sample_rejected_delegate` on `sample_rejected`

3.14.2.13 `subscription_matched_delegate` on `subscription_matched`

The documentation for this class was generated from the following file:

- `/home/ctucker/coredx_v3.4rc/src/dds_csharp/base.cs`

## 3.15 DomainParticipantQos Class Reference

Structure that holds [DomainParticipant](#) Quality of Service policies.

### Public Attributes

- [UserDataQosPolicy](#) `user_data`
- [EntityFactoryQosPolicy](#) `entity_factory`

### 3.15.1 Detailed Description

Structure that holds [DomainParticipant](#) Quality of Service policies.

#### See also

[DomainParticipant::set\\_qos\(\)](#)  
[DomainParticipant::get\\_qos\(\)](#)  
[DomainParticipantFactory::create\\_participant\(\)](#)  
[DomainParticipantFactory::set\\_default\\_participant\\_qos\(\)](#)  
[DomainParticipantFactory::get\\_default\\_participant\\_qos\(\)](#)

### 3.15.2 Member Data Documentation

#### 3.15.2.1 EntityFactoryQosPolicy `entity_factory`

Controls the behavior of the [DomainParticipant](#) `create` operations.

#### 3.15.2.2 UserDataQosPolicy `user_data`

A sequence of octets associated with a [DomainParticipant](#).

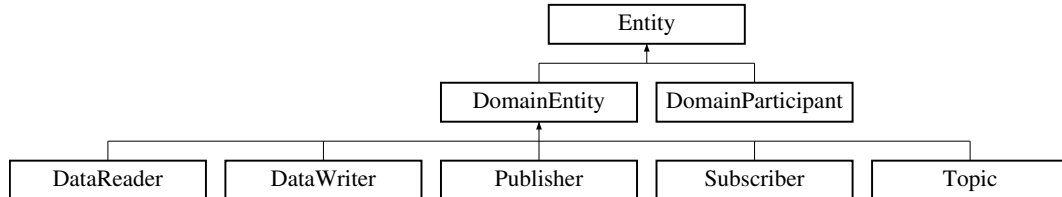
The documentation for this class was generated from the following file:

- `/home/ctucker/coredx_v3.4rc/src/dds_csharp/base.cs`

## 3.16 Entity Class Reference

Base class for all DDS Entities.

Inheritance diagram for Entity:



### Public Member Functions

- virtual [StatusCondition](#) `get_statuscondition` ()
- virtual `uint` `get_status_changes` ()
- virtual `ReturnCode_t` `enable` ()
- virtual `InstanceHandle_t` `get_instance_handle` ()

#### 3.16.1 Detailed Description

Base class for all DDS Entities.

#### 3.16.2 Member Function Documentation

##### 3.16.2.1 virtual `ReturnCode_t` `enable` ( ) [`inline`, `virtual`]

Enable this [Entity](#). An [Entity](#) will begin to communicate only after it is enabled.

Reimplemented in [DomainParticipant](#), [DataReader](#), [DataWriter](#), [Publisher](#), and [Subscriber](#).

##### 3.16.2.2 virtual `InstanceHandle_t` `get_instance_handle` ( ) [`inline`, `virtual`]

Gets the handle that locally identifies this [Entity](#).

Reimplemented in [DomainParticipant](#), [DataReader](#), [DataWriter](#), [Publisher](#), and [Subscriber](#).

##### 3.16.2.3 virtual `uint` `get_status_changes` ( ) [`inline`, `virtual`]

Gets the current status changes from this [Entity](#).

### 3.16.2.4 virtual `StatusCondition` `get_statuscondition ( )` [`inline`, `virtual`]

Gets the [StatusCondition](#) associated with this [Entity](#).

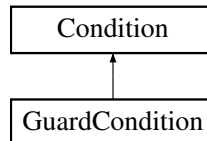
The documentation for this class was generated from the following file:

- `/home/ctucker/coredx_v3.4rc/src/dds_csharp/entity.cs`

## 3.17 GuardCondition Class Reference

A [GuardCondition](#) is a [Condition](#) where the **trigger\_value** is under application control.

Inheritance diagram for GuardCondition:



### Public Member Functions

- [GuardCondition](#) ()

#### 3.17.1 Detailed Description

A [GuardCondition](#) is a [Condition](#) where the **trigger\_value** is under application control.

#### 3.17.2 Constructor & Destructor Documentation

##### 3.17.2.1 [GuardCondition](#) ( ) [`inline`]

This routine creates a [GuardCondition](#).

The documentation for this class was generated from the following file:

- `/home/ctucker/coredx_v3.4rc/src/dds_csharp/cond.cs`

## 3.18 InconsistentTopicStatus Class Reference

Status related to the `on_inconsistent_topic` listener methods of the [TopicListener](#) structure.

### Public Attributes

- int [total\\_count](#)  
*Cummulative count of the discovered Topics having a matching name and inconsistent characteristics.*
- int [total\\_count\\_change](#)  
*Change in **total\_count** since the last time the listener was called or status was read.*

### 3.18.1 Detailed Description

Status related to the `on_inconsistent_topic` listener methods of the [TopicListener](#) structure.

The documentation for this class was generated from the following file:

- `/home/ctucker/coredx_v3.4rc/src/dds_csharp/base.cs`



## 3.19 LivelinessChangedStatus Class Reference

Status related to the `on_liveliness_changed` listener methods of the [DataReader](#), [Subscriber](#), and [DomainParticipant](#) structures.

### Public Attributes

- int [alive\\_count](#)  
*The number of 'active' DataWriters matched to this [DataReader](#).*
- int [not\\_alive\\_count](#)  
*The number of 'not-alive' DataWriters matched to this [DataReader](#).*
- int [alive\\_count\\_change](#)  
*Change in [alive\\_count](#) since the last time the listener was called or status was read.*
- int [not\\_alive\\_count\\_change](#)  
*Change in [not\\_alive\\_count](#) since the last time the listener was called or status was read.*
- InstanceHandle\_t [last\\_publication\\_handle](#)  
*Handle identifying the most recent [DataWriter](#) whose liveliness changed.*

### 3.19.1 Detailed Description

Status related to the `on_liveliness_changed` listener methods of the [DataReader](#), [Subscriber](#), and [DomainParticipant](#) structures.

The documentation for this class was generated from the following file:

- `/home/ctucker/coredx_v3.4rc/src/dds_csharp/base.cs`

## 3.20 LivelinessLostStatus Class Reference

Status related to the `on_liveliness_lost` listener methods of the [DataWriter](#), [Publisher](#), and [DomainParticipant](#) structures.

### Public Attributes

- int [total\\_count](#)  
*Cummulative number of times that an 'alive' [DataWriter](#) became not alive.*
- int [total\\_count\\_change](#)  
*Change in **total\_count** since the last time the listener was called or status was read.*

### 3.20.1 Detailed Description

Status related to the `on_liveliness_lost` listener methods of the [DataWriter](#), [Publisher](#), and [DomainParticipant](#) structures.

The documentation for this class was generated from the following file:

- `/home/ctucker/coredx_v3.4rc/src/dds_csharp/base.cs`

## 3.21 OfferedDeadlineMissedStatus Class Reference

Status related to the `on_offered_deadline_missed` listener methods of the [DataWriter](#), [Publisher](#), and [Domain-Participant](#) structures.

### Public Attributes

- int `total_count`  
*Cummulative count of the number of deadlines missed by this [DataWriter](#).*
- int `total_count_change`  
*Change in **total\_count** since the last time the listener was called or status was read.*
- InstanceHandle\_t `last_instance_handle`  
*Handle identifying the most recent instance whose deadline was missed.*

### 3.21.1 Detailed Description

Status related to the `on_offered_deadline_missed` listener methods of the [DataWriter](#), [Publisher](#), and [Domain-Participant](#) structures.

The documentation for this class was generated from the following file:

- `/home/ctucker/coredx_v3.4rc/src/dds_csharp/base.cs`

## 3.22 OfferedIncompatibleQosStatus Class Reference

Status related to the `on_offered_incompatible_qos` listener methods of the [DataWriter](#), [Publisher](#), and [DomainParticipant](#) structures.

### Public Attributes

- int [total\\_count](#)  
*Cummulative count of the number of DataWriters discovered having matching [Topic](#) and incompatible QoS.*
- int [total\\_count\\_change](#)  
*Change in **total\_count** since the last time the listener was called or status was read.*
- [QosPolicyId\\_t](#) [last\\_policy\\_id](#)  
*Id of the most recent requested incompatible QoS policy.*
- [QosPolicyCount\[\]](#) [policies](#)  
*A list of QoS policies and the total number of times each QoS policy was found to be incompatible.*

### 3.22.1 Detailed Description

Status related to the `on_offered_incompatible_qos` listener methods of the [DataWriter](#), [Publisher](#), and [DomainParticipant](#) structures.

The documentation for this class was generated from the following file:

- `/home/ctucker/coredx_v3.4rc/src/dds_csharp/base.cs`

## 3.23 PublicationMatchedStatus Class Reference

Status related to the `on_publication_matched` listener methods of the [DataWriter](#), [Publisher](#), and [DomainParticipant](#) structures.

### Public Attributes

- int `total_count`  
*Cummulative count of the number of times this [DataWriter](#) has discovered a matching [DataReader](#).*
- int `total_count_change`  
*Change in **total\_count** since the last time the listener was called or status was read.*
- int `current_count`  
*The current number of [DataReaders](#) matched to the [DataWriter](#).*
- int `current_count_change`  
*Change in **current\_count** since the last time the listener was called or status was read.*

### 3.23.1 Detailed Description

Status related to the `on_publication_matched` listener methods of the [DataWriter](#), [Publisher](#), and [DomainParticipant](#) structures.

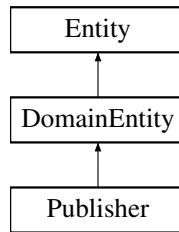
The documentation for this class was generated from the following file:

- `/home/ctucker/coredx_v3.4rc/src/dds_csharp/base.cs`

## 3.24 Publisher Class Reference

The [Publisher](#) configures, creates, manages and destroys DataWriters.

Inheritance diagram for Publisher:



### Public Member Functions

- override [ReturnCode\\_t enable](#) ()
- override [InstanceHandle\\_t get\\_instance\\_handle](#) ()
- [DomainParticipant get\\_participant](#) ()
- [DataWriter create\\_datawriter](#) ([Topic](#) topic, [DataWriterQos](#) dw\_qos, [DataWriterListener](#) listener, uint mask)
- [ReturnCode\\_t delete\\_datawriter](#) ([DataWriter](#) datawriter)
- [ReturnCode\\_t delete\\_contained\\_entities](#) ()
- [DataWriter lookup\\_datawriter](#) ([String](#) topic\_name)
- [ReturnCode\\_t set\\_qos](#) ([PublisherQos](#) qos)
- [ReturnCode\\_t get\\_qos](#) ([PublisherQos](#) qos)
- [ReturnCode\\_t set\\_listener](#) ([PublisherListener](#) new\_listener, uint mask)
- [PublisherListener get\\_listener](#) ()
- [ReturnCode\\_t suspend\\_publications](#) ()
- [ReturnCode\\_t resume\\_publications](#) ()
- [ReturnCode\\_t begin\\_coherent\\_changes](#) ()
- [ReturnCode\\_t end\\_coherent\\_changes](#) ()
- [ReturnCode\\_t wait\\_for\\_acknowledgments](#) ([Duration\\_t](#) max\_wait)
  - *Block until all writers contained by this publisher have received acknowledgements.*
- [ReturnCode\\_t set\\_default\\_datawriter\\_qos](#) ([DataWriterQos](#) qos)
- [ReturnCode\\_t get\\_default\\_datawriter\\_qos](#) ([DataWriterQos](#) qos)
- [ReturnCode\\_t copy\\_from\\_topic\\_qos](#) ([DataWriterQos](#) qos, [TopicQos](#) topic\_qos)

### 3.24.1 Detailed Description

The [Publisher](#) configures, creates, manages and destroys DataWriters.

## 3.24.2 Member Function Documentation

### 3.24.2.1 `ReturnCode_t begin_coherent_changes ( ) [inline]`

#### Not Yet Supported

This operation is not yet implemented.

### 3.24.2.2 `ReturnCode_t copy_from_topic_qos ( DataWriterQos qos, TopicQos topic_qos ) [inline]`

This operation copies the QoS settings in **a\_topic\_qos** to the corresponding settings in **a\_datawriter\_qos**. The **a\_datawriter\_qos** parameter is populated with a copy of the QoS policies from the **a\_topic\_qos** structure. QoS entries in the datawriter qos structure will be overwritten with the values from the topic.

### 3.24.2.3 `DataWriter create_datawriter ( Topic topic, DataWriterQos dw_qos, DataWriterListener listener, uint mask ) [inline]`

This operation creates a [DataWriter](#). The created [DataWriter](#) is contained within the [Publisher p](#). It is associated with the [Topic](#), [ContentFilteredTopic](#), or [MultiTopic](#) indicated by **a\_topic**, and has the [DataWriterQos](#) indicated by **qos**. The **qos** argument may be passed `DDS.DATAWRITER_QOS_DEFAULT`, which indicates that the [Publisher](#) should use its currently configured default data writer QoS values. The [DataWriterListener](#) **a\_listener**, is installed at creation time.

The created [DataWriter](#) (if not NULL) must be destroyed by a call to [Publisher.delete\\_datawriter\(\)](#).

This routine will fail if the provided QoS settings are internally inconsistent. In this case, the routine will return `NULL`.

### 3.24.2.4 `ReturnCode_t delete_contained_entities ( ) [inline]`

This operation deletes all the [DataWriters](#) created by means of the [Publisher.create\\_datawriter\(\)](#) operation on the [Publisher p](#). This routine will recursively call the corresponding [delete\\_contained\\_entities\(\)](#) operation on each of the contained [DataWriter](#) objects. After successful execution, the application may delete the [Publisher](#) by calling [DomainParticipant.delete\\_publisher\(\)](#).

If any of the objects cannot be deleted, this routine will return `ReturnCode_t.RETCODE_PRECONDITION_NOT_MET`.

### 3.24.2.5 `ReturnCode_t delete_datawriter ( DataWriter datawriter ) [inline]`

This operation deletes a [DataWriter](#). If the provided [DataWriter](#) **a\_datawriter** was not created by [Publisher p](#), the routine will fail and will return `ReturnCode_t.RETCODE_PRECONDITION_NOT_MET`.

### 3.24.2.6 override ReturnCode\_t enable ( ) [inline, virtual]

Enables the [Publisher](#). A [Publisher](#) is created either enabled or not based on the [DomainParticipantQos](#) setting [entity\\_factory](#). When a [Publisher](#) is not enabled, only the following sub-set of all [Publisher](#) operations are legal:

- operations to get and set QoS policies,
- factory operations (create, delete),
- [get\\_statuscondition\(\)](#),
- [get\\_status\\_changes\(\)](#),
- lookup operations

Any other operation may return the `ReturnCode_t.RETCODE_NOT_ENABLED` error. `Publisher_enable()` may be called on an already enabled [Subscriber](#) [it will have no effect].

Reimplemented from [Entity](#).

### 3.24.2.7 ReturnCode\_t end\_coherent\_changes ( ) [inline]

#### Not Yet Supported

This operation is not yet implemented.

### 3.24.2.8 ReturnCode\_t get\_default\_datawriter\_qos ( DataWriterQos qos ) [inline]

Provides access to the default [DataWriterQos](#) settings held in the [Publisher](#) `p`. The provided `qos` argument is populated with the current default qos settings.

### 3.24.2.9 override InstanceHandle\_t get\_instance\_handle ( ) [inline, virtual]

Gets the handle that locally identifies this [Entity](#).

Reimplemented from [Entity](#).

### 3.24.2.10 PublisherListener get\_listener ( ) [inline]

This operation returns the currently installed [PublisherListener](#).

### 3.24.2.11 DomainParticipant get\_participant ( ) [inline]

This operation returns the [DomainParticipant](#) this [Publisher](#) belongs to.



**3.24.2.12 ReturnCode\_t get\_qos ( PublisherQos qos ) [inline]**

Returns the current [PublisherQos](#) settings held in the [Publisher p](#). The **qos** parameter is populated with a copy of the current [Publisher](#) QoS properties.

**3.24.2.13 DataWriter lookup\_datawriter ( String topic\_name ) [inline]**

This operation retrieves a previously-created [DataWriter](#) contained in the [Publisher](#), attached to a [Topic](#) named **topic\_name**. If multiple [DataWriters](#) are found, one of them will be returned. If no matching [DataWriter](#) is found, this routine will return **NULL**.

**3.24.2.14 ReturnCode\_t resume\_publications ( ) [inline]****Not Yet Supported**

This operation is not yet implemented.

**3.24.2.15 ReturnCode\_t set\_default\_datawriter\_qos ( DataWriterQos qos ) [inline]**

Sets the default [DataWriterQos](#) held in the [Publisher](#). This default qos will be used during subsequent calls to [Publisher.create\\_datawriter\(\)](#) if the special [DDS.DATAWRITER\\_QOS\\_DEFAULT](#) value is provided for **qos**. This routine may fail if the provided **qos** argument is not internally consistent. In this case, [ReturnCode\\_t.RETCODE\\_INCONSISTENT\\_POLICY](#) will be returned, and no changes will be made to the [Publisher](#).

**3.24.2.16 ReturnCode\_t set\_listener ( PublisherListener new\_listener, uint mask ) [inline]**

Installs a [PublisherListener](#) on [Publisher p](#). Only one listener may be attached to a [Publisher](#) at a time. A call to [set\\_listener\(\)](#) will replace any current listener with **a\_listener**.

**a\_listener** can be **NULL**, which indicates a listener that does nothing.

**3.24.2.17 ReturnCode\_t set\_qos ( PublisherQos qos ) [inline]**

Sets the [PublisherQos](#) values. These QoS values affect the behavior of the [Publisher](#). This routine may fail if the provided **qos** argument is not internally consistent. In this case, [ReturnCode\\_t.RETCODE\\_INCONSISTENT\\_POLICY](#) will be returned, and no changes will be made to the [Publisher](#) QoS.

**3.24.2.18 ReturnCode\_t suspend\_publications ( ) [inline]****Not Yet Supported**

This operation is not yet implemented.

**3.24.2.19 ReturnCode\_t wait\_for\_acknowledgments ( Duration\_t *max\_wait* ) [inline]**

Block until all writers contained by this publisher have received acknowledgements.

This routine will block until all data written by contained writers has been acknowledged, or until the 'max\_wait' duration has passed. 'max\_wait' can be set to INFINITE, in which case this routine may block indefinitely.

**Return values**

*DDS\_RETCODE\_TIME\_OUT* returned if 'max\_wait' passes before all acks are received

*DDS\_RETCODE\_OK* returned if all acks have been received before 'max\_wait'

The documentation for this class was generated from the following file:

- /home/ctucker/coredx\_v3.4rc/src/dds\_csharp/pub.cs

## 3.25 PublisherListener Class Reference

The [PublisherListener](#) provides asynchronous notification of [Publisher](#) events.

### Public Attributes

- [offered\\_deadline\\_missed\\_delegate](#) [on\\_offered\\_deadline\\_missed](#)
- [offered\\_incompatible\\_qos\\_delegate](#) [on\\_offered\\_incompatible\\_qos](#)
- [liveliness\\_lost\\_delegate](#) [on\\_liveliness\\_lost](#)
- [publication\\_matched\\_delegate](#) [on\\_publication\\_matched](#)

### 3.25.1 Detailed Description

The [PublisherListener](#) provides asynchronous notification of [Publisher](#) events. This listener can be installed during [Publisher](#) creation [DomainParticipant.create\\_publisher\(\)](#), as well as by calling [Publisher.set\\_listener\(\)](#).

#### Note

The listener callback methods should be lightweight and should not block. If a callback method blocks, it will block all other callback operations within the same [DomainParticipant](#).

### 3.25.2 Member Data Documentation

**3.25.2.1** [liveliness\\_lost\\_delegate](#) [on\\_liveliness\\_lost](#)

**3.25.2.2** [offered\\_deadline\\_missed\\_delegate](#) [on\\_offered\\_deadline\\_missed](#)

**3.25.2.3** [offered\\_incompatible\\_qos\\_delegate](#) [on\\_offered\\_incompatible\\_qos](#)

**3.25.2.4** [publication\\_matched\\_delegate](#) [on\\_publication\\_matched](#)

The documentation for this class was generated from the following file:

- `/home/ctucker/coredx_v3.4rc/src/dds_csharp/base.cs`

## 3.26 PublisherQos Class Reference

Structure that holds [Publisher](#) Quality of Service policies.

### Public Attributes

- [PresentationQosPolicy](#) presentation
- [PartitionQosPolicy](#) partition
- [GroupDataQosPolicy](#) group\_data
- [EntityFactoryQosPolicy](#) entity\_factory

### 3.26.1 Detailed Description

Structure that holds [Publisher](#) Quality of Service policies.

#### See also

[Publisher::set\\_qos\(\)](#)  
[Publisher::get\\_qos\(\)](#)  
[DomainParticipant::create\\_publisher\(\)](#)  
[DomainParticipant::set\\_default\\_publisher\\_qos\(\)](#)  
[DomainParticipant::get\\_default\\_publisher\\_qos\(\)](#)

### 3.26.2 Member Data Documentation

#### 3.26.2.1 EntityFactoryQosPolicy entity\_factory

Controls the behavior of the `Publisher_create_datawriter()` operation.

#### 3.26.2.2 GroupDataQosPolicy group\_data

A sequence of octets associated with the [Publisher](#).

#### 3.26.2.3 PartitionQosPolicy partition

Establishes a logical data partition. DataWriters and DataReaders that are 'in' the same partition (ie, the partition of the containing [Publisher](#) and [Subscriber](#) match) can communicate. If the partitions do not match, then they cannot communicate.

### 3.26.2.4 PresentationQosPolicy presentation

Controls the presentation of groups of changes.

#### See also

[Publisher::begin\\_coherent\\_changes\(\)](#) begin\_coherent\_changes()  
[Publisher::end\\_coherent\\_changes\(\)](#) end\_coherent\_changes()  
[Subscriber::begin\\_access\(\)](#) begin\_access()  
[Subscriber::end\\_access\(\)](#) end\_access()

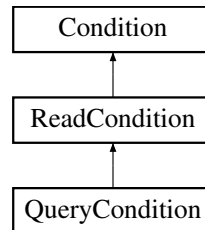
The documentation for this class was generated from the following file:

- /home/ctucker/coredx\_v3.4rc/src/dds\_csharp/base.cs

## 3.27 QueryCondition Class Reference

The **trigger\_value** is driven by the data available, after applying the filter, in the associated [DataReader](#).

Inheritance diagram for QueryCondition:



### Public Member Functions

- [String get\\_query\\_expression \(\)](#)  
*Provides access to the query expression.*
- [ReturnCode\\_t get\\_query\\_parameters \(List< String > qparams\)](#)  
*Provides access to the parameters of the query expression.*
- [ReturnCode\\_t set\\_query\\_parameters \(List< String > parameters\)](#)  
*Modifies the parameters of the query expression.*

### 3.27.1 Detailed Description

The **trigger\_value** is driven by the data available, after applying the filter, in the associated [DataReader](#). CoreDX DDS fully supports QueryConditions as an argument to `DataReader::read_w_condition()` and `DataReader::take_w_conditions()`

#### See also

[DataReader::create\\_querycondition\(\)](#)  
[FooDataReader::read\\_w\\_condition\(\)](#)  
[FooDataReader::take\\_w\\_condition\(\)](#)

#### Not Yet Supported

CoreDX DDS does not yet support QueryConditions as triggers for a [WaitSet](#).

## 3.27.2 Member Function Documentation

### 3.27.2.1 String get\_query\_expression ( ) [inline]

Provides access to the query expression.

The query expression is an SQL syntax conditional expression that is provided when the [QueryCondition](#) is created.

#### See also

[DataReader::create\\_querycondition\(\)](#)

### 3.27.2.2 ReturnCode\_t get\_query\_parameters ( List< String > *qparams* ) [inline]

Provides access to the parameters of the query expression.

The query expression is an SQL syntax conditional expression that is provided when the [QueryCondition](#) is created. The query expression may contain references to positional parameters of the form '0', '1'. These parameters can be changed dynamically to affect the expression.

#### See also

[DataReader::create\\_querycondition\(\)](#)  
[QueryCondition::set\\_query\\_parameters\(\)](#)

### 3.27.2.3 ReturnCode\_t set\_query\_parameters ( List< String > *parameters* ) [inline]

Modifies the parameters of the query expression.

The **query\_expression** is an SQL like condition expression, and the **parameters** argument provides optional parameters that are referenced by the **query\_expression**. The syntax for referring to parameters in a query\_expression is the percent sign " " followed by a number. The number is the index of the parameter in the **query\_parameters** sequence. Parameters are counted starting at zero. So, "%0" refers to the first parameter, and "%4" refers to the fifth parameter. Using this syntax, the expression "x<%0" would test the value of 'x' against the first parameter in the sequence.

This routine allows the parameters to be changed dynamically.

#### See also

[DataReader::create\\_querycondition\(\)](#)

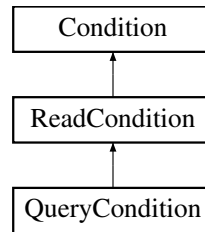
The documentation for this class was generated from the following file:

- /home/ctucker/coredx\_v3.4rc/src/dds\_csharp/cond.cs

## 3.28 ReadCondition Class Reference

A [ReadCondition](#) is a specialized [Condition](#) associated with a [DataReader](#).

Inheritance diagram for ReadCondition:



### Public Member Functions

- [uint get\\_sample\\_state\\_mask\(\)](#)
- [uint get\\_view\\_state\\_mask\(\)](#)
- [uint get\\_instance\\_state\\_mask\(\)](#)
- [DataReader get\\_datareader\(\)](#)

### 3.28.1 Detailed Description

A [ReadCondition](#) is a specialized [Condition](#) associated with a [DataReader](#). The **trigger\_value** is driven by the data available in the associated [DataReader](#). A [ReadCondition](#) is obtained by calling the `DataReader::create_readcondition()` function. When the [ReadCondition](#) is no longer needed, it should be destroyed by a call to `DataReader::delete_readcondition()`.

#### See also

`DataReader::create_readcondition(long sample_states, long view_states, long instance_states)`  
[DataReader::delete\\_readcondition\(ReadCondition\)](#)

### 3.28.2 Member Function Documentation

#### 3.28.2.1 DataReader get\_datareader() [inline]

Gets the single [DataReader](#) associated with this [ReadCondition](#).

#### 3.28.2.2 uint get\_instance\_state\_mask() [inline]

Gets the current value of the `instance_state_mask` in this [ReadCondition](#).



**3.28.2.3 uint get\_sample\_state\_mask ( ) [inline]**

Gets the current value of the sample\_state\_mask in this [ReadCondition](#).

**3.28.2.4 uint get\_view\_state\_mask ( ) [inline]**

Gets the current value of the view\_state\_mask in this [ReadCondition](#).

The documentation for this class was generated from the following file:

- /home/ctucker/coredx\_v3.4rc/src/dds\_csharp/cond.cs

## 3.29 RequestedDeadlineMissedStatus Class Reference

Status related to the `on_requested_deadline_missed` listener methods of the [DataReader](#), [Subscriber](#), and [DomainParticipant](#) structures.

### Public Attributes

- int [total\\_count](#)  
*Cummulative count of the number of detected deadline misses for any instance read by the [DataReader](#).*
- int [total\\_count\\_change](#)  
*Change in **total\_count** since the last time the listener was called or status was read.*
- InstanceHandle\_t [last\\_instance\\_handle](#)  
*Handle identifying the most recent instance whose deadline was missed.*

### 3.29.1 Detailed Description

Status related to the `on_requested_deadline_missed` listener methods of the [DataReader](#), [Subscriber](#), and [DomainParticipant](#) structures.

The documentation for this class was generated from the following file:

- `/home/ctucker/coredx_v3.4rc/src/dds_csharp/base.cs`

## 3.30 RequestedIncompatibleQoSStatus Class Reference

Status related to the `on_requested_incompatible_qos` listener methods of the [DataReader](#), [Subscriber](#), and [DomainParticipant](#) structures.

### Public Attributes

- `int total_count`  
*Cummulative count of the number of discovered DataReaders having matching [Topic](#) and incompatible QoS.*
- `int total_count_change`  
*Change in **total\_count** since the last time the listener was called or status was read.*
- `QoSPolicyId_t last_policy_id`  
*Handle identifying the most recent QoS policy detected to be incompatible.*
- `QoSPolicyCount[] policies`  
*A list of QoS policies and the total number of times each QoS policy was found to be incompatible.*

### 3.30.1 Detailed Description

Status related to the `on_requested_incompatible_qos` listener methods of the [DataReader](#), [Subscriber](#), and [DomainParticipant](#) structures.

The documentation for this class was generated from the following file:

- `/home/ctucker/coredx_v3.4rc/src/dds_csharp/base.cs`

## 3.31 SampleInfo Class Reference

The [SampleInfo](#) structure contains information associated with each Sample. The `DataReader.read()` and `take()` operations return two vectors. One vector contains Sample(s) and the other contains SampleInfo(s). There is a one-to-one correspondence between items in these two vectors. Each Sample is described by the corresponding [SampleInfo](#) instance.

### Public Attributes

- uint [sample\\_state](#)
- uint [view\\_state](#)
- uint [instance\\_state](#)
- [Time\\_t](#) [source\\_timestamp](#)
- [Time\\_t](#) [reception\\_timestamp](#)
- [InstanceHandle\\_t](#) [instance\\_handle](#)
- [InstanceHandle\\_t](#) [publication\\_handle](#)
- int [disposed\\_generation\\_count](#)
- int [no\\_writers\\_generation\\_count](#)
- int [sample\\_rank](#)
- int [generation\\_rank](#)
- int [absolute\\_generation\\_rank](#)
- bool [valid\\_data](#)

### 3.31.1 Detailed Description

The [SampleInfo](#) structure contains information associated with each Sample. The `DataReader.read()` and `take()` operations return two vectors. One vector contains Sample(s) and the other contains SampleInfo(s). There is a one-to-one correspondence between items in these two vectors. Each Sample is described by the corresponding [SampleInfo](#) instance.

### 3.31.2 Member Data Documentation

#### 3.31.2.1 `int absolute_generation_rank`

The generation difference between this sample and the most recent sample. The **`absolute_generation_rank`** indicates the generation difference (ie, the number of times the instance was disposed and became alive again) between this sample, and the most recent sample (possibly not in the returned collection) of this instance.

#### 3.31.2.2 `int disposed_generation_count`

The number of times the instance has become 'ALIVE' after being explicitly disposed. (Computed at the time the sample arrives at the [DataReader](#).)

### 3.31.2.3 int generation\_rank

The generation difference of this sample and the most recent sample in the collection. **generation\_rank** indicates the generation difference (ie, the number of times the instance was disposed and became alive again) between this sample, and the most recent sample in the collection related to this instance.

### 3.31.2.4 InstanceHandle\_t instance\_handle

The handle that locally identifies the associated instance.

### 3.31.2.5 uint instance\_state

Indicates whether the associated instance currently exists. **instance\_state** can be one of:

[DDS.ALIVE\\_INSTANCE\\_STATE](#)

[DDS.NOT\\_ALIVE\\_DISPOSED\\_INSTANCE\\_STATE](#)

[DDS.NOT\\_ALIVE\\_NO\\_WRITERS\\_INSTANCE\\_STATE](#)

Can use [DDS.NOT\\_ALIVE\\_INSTANCE\\_STATE](#) as a test for either 'NOT\_ALIVE' state. For example if (sample\_info->view\_state & [DDS.NOT\\_ALIVE\\_INSTANCE\\_STATE](#)) ...

### 3.31.2.6 int no\_writers\_generation\_count

The number of times the instance has become 'ALIVE' after being automatically disposed due to no active writers. (Computed at the time the sample arrives at the [DataReader](#).)

### 3.31.2.7 InstanceHandle\_t publication\_handle

The local handle of the source [DataWriter](#).

### 3.31.2.8 Time\_t reception\_timestamp

The time when the sample was received by the [DataReader](#).

### 3.31.2.9 int sample\_rank

Number of samples related to this instances that follow in the collection returned by the [DataReader](#) **read** or **take** operations.

### 3.31.2.10 uint sample\_state

The associated data sample has/has not been read previously.

**sample\_state** indicates whether or not the [DataReader](#) has previously read the associated sample.

One of:

[DDS.READ\\_SAMPLE\\_STATE](#)

[DDS.NOT\\_READ\\_SAMPLE\\_STATE](#).

Use [DDS.ANY\\_SAMPLE\\_STATE](#) to test for either state.

### 3.31.2.11 Time\_t source\_timestamp

The time provided by the [DataWriter](#) when the sample was written.

### 3.31.2.12 bool valid\_data

Is set to true if the associated DataSample contains data. The associated DataSample may not contain data if it this sample indicates a change in sample state (for example ALIVE -> DISPOSED).

### 3.31.2.13 uint view\_state

Associated instance has/has not been seen before. **view\_state** indicates whether the [DataReader](#) has already seen samples for the most current generation of the related instance.

**view\_state** can be one of:

[DDS.NEW\\_VIEW\\_STATE](#)

[DDS.NOT\\_NEW\\_VIEW\\_STATE](#)

The documentation for this class was generated from the following file:

- /home/ctucker/coredx\_v3.4rc/src/dds\_csharp/si.cs

## 3.32 SampleLostStatus Class Reference

Status related to the on\_sample\_lost listener methods of the [DataReader](#), [Subscriber](#), and [DomainParticipant](#) structures.

### Public Attributes

- int [total\\_count](#)  
*Cummulative count of all samples lost under the [Topic](#).*
- int [total\\_count\\_change](#)  
*Change in **total\_count** since the last time the listener was called or status was read.*

### 3.32.1 Detailed Description

Status related to the on\_sample\_lost listener methods of the [DataReader](#), [Subscriber](#), and [DomainParticipant](#) structures.

The documentation for this class was generated from the following file:

- `/home/ctucker/coredx_v3.4rc/src/dds_csharp/base.cs`

### 3.33 SampleRejectedStatus Class Reference

Status related to the `on_sample_rejected` listener methods of the [DataReader](#), [Subscriber](#), and [DomainParticipant](#) structures.

#### Public Attributes

- int [total\\_count](#)  
*Cummulative count of samples rejected by the [DataReader](#).*
- int [total\\_count\\_change](#)  
*Change in **total\_count** since the last time the listener was called or status was read.*
- [SampleRejectedStatusKind](#) [last\\_reason](#)  
*The reason for rejecting the most recently rejected sample.*
- [InstanceHandle\\_t](#) [last\\_instance\\_handle](#)  
*The handle of the instance associated with the most recently rejected sample.*

#### 3.33.1 Detailed Description

Status related to the `on_sample_rejected` listener methods of the [DataReader](#), [Subscriber](#), and [DomainParticipant](#) structures.

The documentation for this class was generated from the following file:

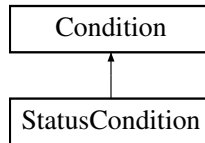
- `/home/ctucker/coredx_v3.4rc/src/dds_csharp/base.cs`



## 3.34 StatusCondition Class Reference

A [StatusCondition](#) is a condition associated with an [Entity](#). The **trigger\_value** is driven by the communication status of the associated [Entity](#).

Inheritance diagram for StatusCondition:



### Public Member Functions

- [ReturnCode\\_t set\\_enabled\\_statuses](#) (uint mask)
- [uint get\\_enabled\\_statuses](#) ()
- [Entity get\\_entity](#) ()

#### 3.34.1 Detailed Description

A [StatusCondition](#) is a condition associated with an [Entity](#). The **trigger\_value** is driven by the communication status of the associated [Entity](#).

#### 3.34.2 Member Function Documentation

##### 3.34.2.1 [uint get\\_enabled\\_statuses](#) ( ) [[inline](#)]

This routine returns the statuses which are enabled in this [StatusCondition](#). The statuses are returned as a bitmask.

See also

[DDS::ALL\\_STATUS](#)

##### 3.34.2.2 [Entity get\\_entity](#) ( ) [[inline](#)]

This routine returns the single entity associated with this [StatusCondition](#).

### 3.34.2.3 ReturnCode\_t set\_enabled\_statuses ( uint *mask* ) [inline]

This routine sets the statuses which are enabled in this [StatusCondition](#). The statuses are provided as a bitmask.

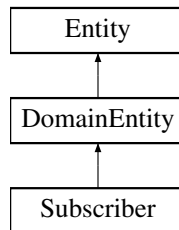
The documentation for this class was generated from the following file:

- /home/ctucker/coredx\_v3.4rc/src/dds\_csharp/cond.cs

## 3.35 Subscriber Class Reference

The [Subscriber](#) configures, creates, manages and destroys DataReaders.

Inheritance diagram for Subscriber:



### Public Member Functions

- override [ReturnCode\\_t enable](#) ()
- override [InstanceHandle\\_t get\\_instance\\_handle](#) ()
- [DataReader create\\_datareader](#) ([TopicDescription](#) topic, [DataReaderQos](#) dr\_qos, [DataReaderListener](#) listener, uint mask)
- [ReturnCode\\_t delete\\_datareader](#) ([DataReader](#) datareader)
- [ReturnCode\\_t delete\\_contained\\_entities](#) ()
- [DataReader lookup\\_datareader](#) ([String](#) topic\_name)
- [ReturnCode\\_t get\\_datareaders](#) ([List](#)< [DataReader](#) > readers, long sample\_states, long view\_states, long instance\_states)
- [DomainParticipant get\\_participant](#) ()
- [ReturnCode\\_t set\\_qos](#) ([SubscriberQos](#) qos)
- [ReturnCode\\_t get\\_qos](#) ([SubscriberQos](#) qos)
- [ReturnCode\\_t set\\_listener](#) ([SubscriberListener](#) new\_listener, uint mask)
- [SubscriberListener get\\_listener](#) ()
- [ReturnCode\\_t begin\\_access](#) ()
- [ReturnCode\\_t end\\_access](#) ()
- [ReturnCode\\_t set\\_default\\_datareader\\_qos](#) ([DataReaderQos](#) qos)
- [ReturnCode\\_t get\\_default\\_datareader\\_qos](#) ([DataReaderQos](#) qos)
- [ReturnCode\\_t copy\\_from\\_topic\\_qos](#) ([DataReaderQos](#) qos, [TopicQos](#) topic\_qos)

### 3.35.1 Detailed Description

The [Subscriber](#) configures, creates, manages and destroys DataReaders.

## 3.35.2 Member Function Documentation

### 3.35.2.1 `ReturnCode_t begin_access ( ) [inline]`

This operation indicates that the application is about to access the data samples in any of the `DataReader` objects contained in the `Subscriber s`. The application is expected to use this operation only if PRESENTATION QoSPolicy of the `Subscriber` has the `access_scope` set to GROUP. [Otherwise this routine has no effect.]

#### Not Yet Supported

This is currently unsupported in the C# language binding.

### 3.35.2.2 `ReturnCode_t copy_from_topic_qos ( DataReaderQos qos, TopicQos topic_qos ) [inline]`

This operation copies the QoS settings in `a_topic_qos` to the corresponding settings in `a_datareader_qos`. The `a_datareader_qos` parameter is populated with a copy of the QoS policies from the `a_topic_qos` structure. QoS entries in the `datareader_qos` structure will be overwritten with the values from the topic.

### 3.35.2.3 `DataReader create_datareader ( TopicDescription topic, DataReaderQos dr_qos, DataReaderListener listener, uint mask ) [inline]`

This operation creates a `DataReader`. The created `DataReader` is contained within the `Subscriber s`. It is associated with the `Topic`, `ContentFilteredTopic`, or `MultiTopic` indicated by `a_topic`, and has the `DataReaderQos` indicated by `qos`. The `qos` argument may be passed `DDS.DATAREADER_QOS_DEFAULT`, which indicates that the `Subscriber` should use its currently configured default data reader QoS values. The `DataReaderListener a_listener`, is installed at creation time.

The created `DataReader` (if not NULL) must be destroyed by a call to `Subscriber.delete_datareader()`.

This routine will fail if the provided QoS settings are internally inconsistent. In this case, the routine will return `NULL`.

### 3.35.2.4 `ReturnCode_t delete_contained_entities ( ) [inline]`

This operation deletes all the `DataReaders` created by means of the `Subscriber.create_datareader()` operation on the `Subscriber s`. This routine will recursively call the corresponding `delete_contained_entities()` operation on each of the contained `DataReader` objects. If successful, this operation will recursively delete all objects contained with this `Subscriber`. After successful execution, the application may delete the `Subscriber` by calling `DomainParticipant.delete_subscriber()`.

If any of the objects cannot be deleted, this routine will return `ReturnCode_t.RETCODE_PRECONDITION_NOT_MET`.

**3.35.2.5 ReturnCode\_t delete\_datareader ( DataReader *datareader* ) [inline]**

This operation deletes a [DataReader](#). If the provided [DataReader](#) **a\_datareader** was not created by [Subscriber](#) s, the routine will fail and will return `ReturnCode_t.RETCODE_PRECONDITION_NOT_MET`. If the indicated [DataReader](#) has any outstanding [ReadCondition](#) or [QueryCondition](#) objects the routine will fail and will return `ReturnCode_t.RETCODE_PRECONDITION_NOT_MET`. If the indicated [DataReader](#) has any outstanding 'loans' (from `read()` or `take()` operations), the routine will fail and will return `ReturnCode_t.RETCODE_PRECONDITION_NOT_MET`.

**3.35.2.6 override ReturnCode\_t enable ( ) [inline, virtual]**

Enables the [Subscriber](#). A [Subscriber](#) is created either enabled or not based on the [DomainParticipantQos](#) setting `entity_factory`. When a [Subscriber](#) is not enabled, only the following sub-set of all [Subscriber](#) operations are legal:

- operations to get and set QoS policies,
- factory operations (create, delete),
- [get\\_statuscondition\(\)](#),
- [get\\_status\\_changes\(\)](#),
- lookup operations

Any other [Subscriber](#) operation may return the `ReturnCode_t.RETCODE_NOT_ENABLED` error. `Subscriber_enable()` may be called on an already enabled [Subscriber](#) [it will have no effect].

Reimplemented from [Entity](#).

**3.35.2.7 ReturnCode\_t end\_access ( ) [inline]**

This operation closes a corresponding [Subscriber.begin\\_access\(\)](#).

**Not Yet Supported**

This is currently unsupported in the C# language binding.

**3.35.2.8 ReturnCode\_t get\_datareaders ( List< DataReader > *readers*, long *sample\_states*, long *view\_states*, long *instance\_states* ) [inline]**

This operation allows the application to access [DataReader](#) objects that contain samples with the specified `sample_states`, `view_states`, and `instance_states`.

If the PRESENTATION QoSPolicy of the [Subscriber](#) to which the [DataReader](#) belongs has the access\_scope set to GROUP, this operation should be invoked only inside a begin\_access/end\_access block. Otherwise it will return the error PRECONDITION\_NOT\_MET.

The returned collection of [DataReader](#) objects may either be a set (containing each [DataReader](#) at most once in no specified order), or a list (containing each [DataReader](#) one or more times in a specific order).

1. If PRESENTATION access\_scope is INSTANCE or TOPIC, the returned collection is a set.
2. If PRESENTATION access\_scope is GROUP and ordered\_access is set to TRUE, then the returned collection is a list.

This difference supports alternate access mechanisms.

### Not Yet Supported

This is currently unsupported in the C# language binding.

#### 3.35.2.9 ReturnCode\_t get\_default\_datareader\_qos ( DataReaderQos qos ) [inline]

Provides access to the default [DataReaderQos](#) settings held in the [Subscriber](#) s. The provided qos argument is populated with the current default qos settings.

#### 3.35.2.10 override InstanceHandle\_t get\_instance\_handle ( ) [inline, virtual]

Gets the handle that locally identifies this [Entity](#).

Reimplemented from [Entity](#).

#### 3.35.2.11 SubscriberListener get\_listener ( ) [inline]

This operation returns the currently installed [SubscriberListener](#).

#### 3.35.2.12 DomainParticipant get\_participant ( ) [inline]

This operation returns the [DomainParticipant](#) this [Subscriber](#) belongs to.

#### 3.35.2.13 ReturnCode\_t get\_qos ( SubscriberQos qos ) [inline]

Returns the current [SubscriberQos](#) settings held in the [Subscriber](#) s. The qos parameter is populated with a copy of the current [Subscriber](#) QoS properties.

**3.35.2.14 DataReader lookup\_datareader ( String topic\_name ) [inline]**

This operation retrieves a previously-created [DataReader](#) contained in the [Subscriber](#), attached to a [Topic](#) named **topic\_name**. If multiple DataReaders are found, one of them will be returned. If no matching [DataReader](#) is found, this routine will return **NULL**.

This routine is useful to obtain access to a particular built-in [DataReader](#).

**3.35.2.15 ReturnCode\_t set\_default\_datareader\_qos ( DataReaderQos qos ) [inline]**

Sets the default [DataReaderQos](#) held in the [Subscriber](#). This default qos will be used during subsequent calls to [Subscriber.create\\_datareader\(\)](#) if the special [DDS.DATAREADER\\_QOS\\_DEFAULT](#) value is provided for **qos**. This routine may fail if the provided **qos** argument is not internally consistent. In this case, [ReturnCode\\_t.RETCODE\\_INCONSISTENT\\_POLICY](#) will be returned, and no changes will be made to the [Subscriber](#).

**3.35.2.16 ReturnCode\_t set\_listener ( SubscriberListener new\_listener, uint mask ) [inline]**

Installs a [SubscriberListener](#) on [Subscriber s](#). Only one listener may be attached to a [Subscriber](#) at a time. A call to [set\\_listener\(\)](#) will replace any current listener with **a\_listener**.

**a\_listener** can be **NULL**, which indicates a listener that does nothing.

**3.35.2.17 ReturnCode\_t set\_qos ( SubscriberQos qos ) [inline]**

Sets the [SubscriberQos](#) values. These QoS values affect the behavior of the [Subscriber](#). This routine may fail if the provided **qos** argument is not internally consistent. In this case, [ReturnCode\\_t.RETCODE\\_INCONSISTENT\\_POLICY](#) will be returned, and no changes will be made to the [Subscriber](#) QoS.

The documentation for this class was generated from the following file:

- /home/ctucker/coredx\_v3.4rc/src/dds\_csharp/sub.cs

## 3.36 SubscriberListener Class Reference

The [SubscriberListener](#) provides asynchronous notification of [Subscriber](#) events.

### Public Attributes

- [requested\\_deadline\\_missed\\_delegate](#) [on\\_requested\\_deadline\\_missed](#)
- [requested\\_incompatible\\_qos\\_delegate](#) [on\\_requested\\_incompatible\\_qos](#)
- [sample\\_rejected\\_delegate](#) [on\\_sample\\_rejected](#)
- [liveliness\\_changed\\_delegate](#) [on\\_liveliness\\_changed](#)
- [data\\_available\\_delegate](#) [on\\_data\\_available](#)
- [subscription\\_matched\\_delegate](#) [on\\_subscription\\_matched](#)
- [sample\\_lost\\_delegate](#) [on\\_sample\\_lost](#)
- [data\\_on\\_readers\\_delegate](#) [on\\_data\\_on\\_readers](#)

### 3.36.1 Detailed Description

The [SubscriberListener](#) provides asynchronous notification of [Subscriber](#) events. This listener can be installed during [Subscriber](#) creation, [DomainParticipant.create\\_subscriber\(\)](#) as well as by calling [Subscriber.set\\_listener\(\)](#).

#### Note

The listener callback methods should be lightweight and should not block. If a callback method blocks, it will block all other callback operations within the same [DomainParticipant](#).



### 3.36.2 Member Data Documentation

3.36.2.1 `data_available_delegate` on `data_available`

3.36.2.2 `data_on_readers_delegate` on `data_on_readers`

3.36.2.3 `liveliness_changed_delegate` on `liveliness_changed`

3.36.2.4 `requested_deadline_missed_delegate` on `requested_deadline_missed`

3.36.2.5 `requested_incompatible_qos_delegate` on `requested_incompatible_qos`

3.36.2.6 `sample_lost_delegate` on `sample_lost`

3.36.2.7 `sample_rejected_delegate` on `sample_rejected`

3.36.2.8 `subscription_matched_delegate` on `subscription_matched`

The documentation for this class was generated from the following file:

- `/home/ctucker/coredx_v3.4rc/src/dds_csharp/base.cs`

## 3.37 SubscriberQos Class Reference

Structure that holds DDS\_Subscriber Quality of Service policies.

### Public Attributes

- [PresentationQosPolicy](#) presentation
- [PartitionQosPolicy](#) partition
- [GroupDataQosPolicy](#) group\_data
- [EntityFactoryQosPolicy](#) entity\_factory

### 3.37.1 Detailed Description

Structure that holds DDS\_Subscriber Quality of Service policies.

#### See also

[Subscriber::set\\_qos\(SubscriberQos\)](#)  
[Subscriber::get\\_qos\(SubscriberQos\)](#)  
[DomainParticipant::create\\_subscriber\(\)](#)  
[DomainParticipant::set\\_default\\_subscriber\\_qos\(\)](#)  
[DomainParticipant::get\\_default\\_subscriber\\_qos\(\)](#)

### 3.37.2 Member Data Documentation

#### 3.37.2.1 EntityFactoryQosPolicy entity\_factory

Controls the behavior of the [Subscriber.create\\_datareader\(\)](#) operation.

#### 3.37.2.2 GroupDataQosPolicy group\_data

A sequence of octets associated with the [Publisher](#).

#### 3.37.2.3 PartitionQosPolicy partition

\* Establishes a logical data partition. DataWriters and DataReaders that are 'in' the same partition (ie, the partition of the containing [Publisher](#) and [Subscriber](#) match) can communicate. If the partitions do not match, then they cannot communicate.

### 3.37.2.4 PresentationQosPolicy presentation

Controls the presentation of groups of changes.

#### See also

[Publisher::begin\\_coherent\\_changes\(\)](#)  
[Publisher::end\\_coherent\\_changes\(\)](#)  
[Subscriber::begin\\_access\(\)](#)  
[Subscriber::end\\_access\(\)](#)

The documentation for this class was generated from the following file:

- `/home/ctucker/coredx_v3.4rc/src/dds_csharp/base.cs`

## 3.38 SubscriptionMatchedStatus Class Reference

Status related to the `on_subscription_matched` listener methods of the [DataReader](#), [Subscriber](#), and [Domain-Participant](#) structures.

### Public Attributes

- int [total\\_count](#)  
*Cummulative count of the number of times this [DataReader](#) has discovered a matching [DataWriter](#).*
- int [total\\_count\\_change](#)  
*Change in **total\_count** since the last time the listener was called or status was read.*
- int [current\\_count](#)  
*The current number of [DataWriters](#) matched to the [DataReader](#).*
- int [current\\_count\\_change](#)  
*Change in **current\_count** since the last time the listener was called or status was read.*

### 3.38.1 Detailed Description

Status related to the `on_subscription_matched` listener methods of the [DataReader](#), [Subscriber](#), and [Domain-Participant](#) structures.

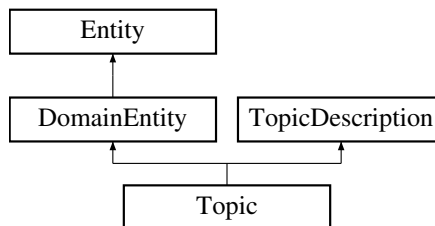
The documentation for this class was generated from the following file:

- `/home/ctucker/coredx_v3.4rc/src/dds_csharp/base.cs`

## 3.39 Topic Class Reference

**Topic** is the basic description of data to be published or subscribed. A topic is identified by a **name** and a **type**. A **Topic** is created by calling `DomainParticipant.create_topic()`. Prior to creating a **Topic**, the associated data type must be registered with the `DomainParticipant` via a call to the `TypeSupportXYZ.register_type()` function. [The `register_type()` function is auto-generated 'type-specific' code.].

Inheritance diagram for **Topic**:



### Public Member Functions

- `DomainParticipant get_participant ()`  
*This operation returns the parent **DomainParticipant** of the **Topic**.*
- `String get_type_name ()`  
*This operation returns **type\_name** of the **Topic**.*
- `String get_name ()`  
*This operation returns **topic\_name** of the **Topic**.*
- `override ReturnCode_t enable ()`
- `override InstanceHandle_t get_instance_handle ()`
- `ReturnCode_t set_qos (TopicQos qos)`
- `ReturnCode_t get_qos (TopicQos qos)`
- `ReturnCode_t set_listener (TopicListener new_listener, uint mask)`
- `TopicListener get_listener ()`
- `ReturnCode_t get_inconsistent_topic_status (out InconsistentTopicStatus status)`

#### 3.39.1 Detailed Description

**Topic** is the basic description of data to be published or subscribed. A topic is identified by a **name** and a **type**. A **Topic** is created by calling `DomainParticipant.create_topic()`. Prior to creating a **Topic**, the associated data type must be registered with the `DomainParticipant` via a call to the `TypeSupportXYZ.register_type()` function. [The `register_type()` function is auto-generated 'type-specific' code.].

## 3.39.2 Member Function Documentation

### 3.39.2.1 override ReturnCode\_t enable ( ) [inline]

Enables the [Topic](#). A [Topic](#) is created either enabled or not based on the [DomainParticipantQos](#) setting [entity\\_factory](#). When a [Topic](#) is not enabled, only the following sub-set of all [Topic](#) operations are legal:

- operations to get and set QoS policies,
- [get\\_name\(\)](#), [get\\_type\\_name\(\)](#), [get\\_participant\(\)](#)
- [get\\_statuscondition\(\)](#),
- [get\\_status\\_changes\(\)](#),

Any other operation may return the `ReturnCode_t.RETURNCODE_NOT_ENABLED` error. [Topic.enable\(\)](#) may be called on an already enabled [Topic](#) [it will have no effect].

### 3.39.2.2 ReturnCode\_t get\_inconsistent\_topic\_status ( out InconsistentTopicStatus status ) [inline]

Provides access to the [InconsistentTopicStatus](#) of the [Topic](#). As a side-effect, this routine will reset the `total_count_change` status field to zero.

### 3.39.2.3 override InstanceHandle\_t get\_instance\_handle ( ) [inline]

Gets the handle that locally identifies this [Entity](#).

### 3.39.2.4 TopicListener get\_listener ( ) [inline]

This operation returns the currently installed [TopicListener](#).

### 3.39.2.5 ReturnCode\_t get\_qos ( TopicQos qos ) [inline]

Returns the current [TopicQos](#) settings held in the [Topic t](#). This routine copies the [Topic](#) QoS properties into `qos`.

### 3.39.2.6 ReturnCode\_t set\_listener ( TopicListener new\_listener, uint mask ) [inline]

Installs a [TopicListener](#) on [Topic t](#). Only one listener may be attached to a [Topic](#) at a time. A call to [set\\_listener\(\)](#) will replace any current listener with `a_listener`.

`a_listener` can be NULL, which indicates a listener that does nothing.

The infrastructure will make an internal copy of the listener structure so that it need not be persisted by the application.

#### 3.39.2.7 `ReturnCode_t set_qos ( TopicQos qos ) [inline]`

Sets the `TopicQos` values. These QoS values affect the behavior of the `Topic`. This routine may fail if the provided `qos` argument is not internally consistent. In this case, `ReturnCode_t.RETCODE_INCONSISTENT_POLICY` will be returned, and no changes will be made to the `Topic` QoS.

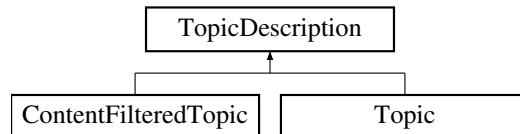
The documentation for this class was generated from the following file:

- `/home/ctucker/coredx_v3.4rc/src/dds_csharp/top.cs`

## 3.40 TopicDescription Interface Reference

[TopicDescription](#) is an interface that provides the foundation for [Topic](#), [ContentFilteredTopic](#), and [Multi-Topic](#).

Inheritance diagram for [TopicDescription](#):



### Public Member Functions

- [DomainParticipant](#) `get_participant ()`  
*This operation returns the parent [DomainParticipant](#) of the [Topic](#).*
- [String](#) `get_type_name ()`  
*This operation returns **type\_name** of the [Topic](#).*
- [String](#) `get_name ()`  
*This operation returns **topic\_name** of the [Topic](#).*

### 3.40.1 Detailed Description

[TopicDescription](#) is an interface that provides the foundation for [Topic](#), [ContentFilteredTopic](#), and [Multi-Topic](#).

The documentation for this interface was generated from the following file:

- `/home/ctucker/coredx_v3.4rc/src/dds_csharp/top.cs`



## 3.41 TopicListener Class Reference

The [TopicListener](#) provides asynchronous notification of [Topic](#) events.

### Public Attributes

- `inconsistent_topic_delegate` [on\\_inconsistent\\_topic](#)

### 3.41.1 Detailed Description

The [TopicListener](#) provides asynchronous notification of [Topic](#) events. This listener can be installed during [Topic](#) creation ([DomainParticipant.create\\_topic\(\)](#) and related) as well as by calling [Topic.set\\_listener\(\)](#).

#### Note

The listener callback methods should be lightweight and should not block. If a callback method blocks, it will block all other callback operations within the same [DomainParticipant](#).

### 3.41.2 Member Data Documentation

#### 3.41.2.1 `inconsistent_topic_delegate` [on\\_inconsistent\\_topic](#)

The documentation for this class was generated from the following file:

- `/home/ctucker/coredx_v3.4rc/src/dds_csharp/base.cs`

## 3.42 TopicQos Class Reference

Structure that holds DDS\_Topic Quality of Service policies.

### Public Attributes

- [TopicDataQosPolicy](#) topic\_data
- [DurabilityQosPolicy](#) durability
- [DurabilityServiceQosPolicy](#) durability\_service
- [DeadlineQosPolicy](#) deadline
- [LatencyBudgetQosPolicy](#) latency\_budget
- [LivelinessQosPolicy](#) liveliness
- [ReliabilityQosPolicy](#) reliability
- [DestinationOrderQosPolicy](#) destination\_order
- [HistoryQosPolicy](#) history
- [ResourceLimitsQosPolicy](#) resource\_limits
- [TransportPriorityQosPolicy](#) transport\_priority
- [LifespanQosPolicy](#) lifespan
- [OwnershipQosPolicy](#) ownership

### 3.42.1 Detailed Description

Structure that holds DDS\_Topic Quality of Service policies.

#### See also

[Topic::set\\_qos\(\)](#)  
[Topic::get\\_qos\(\)](#)  
[DomainParticipant::create\\_topic\(\)](#)  
[DomainParticipant::set\\_default\\_topic\\_qos\(\)](#)

## 3.42.2 Member Data Documentation

- 3.42.2.1 `DeadlineQosPolicy` `deadline`
- 3.42.2.2 `DestinationOrderQosPolicy` `destination_order`
- 3.42.2.3 `DurabilityQosPolicy` `durability`
- 3.42.2.4 `DurabilityServiceQosPolicy` `durability_service`
- 3.42.2.5 `HistoryQosPolicy` `history`
- 3.42.2.6 `LatencyBudgetQosPolicy` `latency_budget`
- 3.42.2.7 `LifespanQosPolicy` `lifespan`
- 3.42.2.8 `LivelinessQosPolicy` `liveliness`
- 3.42.2.9 `OwnershipQosPolicy` `ownership`
- 3.42.2.10 `ReliabilityQosPolicy` `reliability`
- 3.42.2.11 `ResourceLimitsQosPolicy` `resource_limits`
- 3.42.2.12 `TopicDataQosPolicy` `topic_data`
- 3.42.2.13 `TransportPriorityQosPolicy` `transport_priority`

The documentation for this class was generated from the following file:

- `/home/ctucker/coredx_v3.4rc/src/dds_csharp/base.cs`

## 3.43 WaitSet Class Reference

A DDS\_WaitSet maintains a set of [Condition](#) objects and allows the application to wait until one or more of them have a **trigger\_value** of TRUE.

### Public Member Functions

- [WaitSet](#) ()
- void [destroy](#) ()
- [ReturnCode\\_t attach\\_condition](#) ([Condition](#) c)
- [ReturnCode\\_t detach\\_condition](#) ([Condition](#) c)
- [ReturnCode\\_t wait](#) (List< [Condition](#) > active\_conditions, [Duration\\_t](#) timeout)
- [ReturnCode\\_t get\\_conditions](#) (List< [Condition](#) > attached\_conditions)

### 3.43.1 Detailed Description

A DDS\_WaitSet maintains a set of [Condition](#) objects and allows the application to wait until one or more of them have a **trigger\_value** of TRUE. Multiple conditions may be attached to a [WaitSet](#).

#### See also

[Condition](#)

### 3.43.2 Constructor & Destructor Documentation

#### 3.43.2.1 [WaitSet](#) ( ) [[inline](#)]

Constructor.

### 3.43.3 Member Function Documentation

#### 3.43.3.1 [ReturnCode\\_t attach\\_condition](#) ( [Condition](#) c ) [[inline](#)]

Adds the condition **c** to the [WaitSet](#). If another thread is currently 'waiting' on the [WaitSet](#), this newly added condition will unblock that thread if its **trigger\_value** is TRUE.

#### 3.43.3.2 [void destroy](#) ( ) [[inline](#)]

Destructor. Releases internal resources associated with the [WaitSet](#). This [WaitSet](#) is destroyed, and must not be used after this call returns.

**3.43.3.3 ReturnCode\_t detach\_condition ( Condition c ) [inline]**

Adds the condition *c* to the [WaitSet](#). If another thread is currently 'waiting' on the [WaitSet](#), this newly added condition will unblock that thread if its **trigger\_value** is TRUE.

**3.43.3.4 ReturnCode\_t get\_conditions ( List< Condition > attached\_conditions ) [inline]**

Retrieves the current list of attached conditions. Populates the **attached\_conditions** sequence.

**3.43.3.5 ReturnCode\_t wait ( List< Condition > active\_conditions, Duration\_t timeout ) [inline]**

Causes the controlling thread to block until an attached condition is triggered or **timeout** elapses.

A return value of DDS\_RETCODE\_OK indicates that one or more of the attached conditions evaluated to TRUE. Those 'active' conditions are included in the 'out' parameter **active\_conditions**.

A return value of DDS\_RETCODE\_TIMEOUT indicates that the timeout period elapsed without any of the conditions evaluating to TRUE.

The documentation for this class was generated from the following file:

- /home/ctucker/coredx\_v3.4rc/src/dds\_csharp/ws.cs



# Chapter 4

## Data Structure Index

### 4.1 Class List

Here are the classes, structs, unions and interfaces with brief descriptions:

<a href="#">Condition</a> (A <a href="#">Condition</a> can be added to a <a href="#">WaitSet</a> to provide synchronous event notification ) . . .	15
<a href="#">ContentFilteredTopic</a> ( <a href="#">ContentFilteredTopic</a> provides a topic that may include data filtered from a related <a href="#">Topic</a> . The <a href="#">ContentFilteredTopic</a> is associated with another un-filtered topic <b>related_topic</b> . It applies a filter to the data of the related topic. If a data sample passes the filter, it will be made available to a <a href="#">DataReader</a> associated with the <a href="#">ContentFilteredTopic</a> )	17
<a href="#">DataReader</a> (The <a href="#">DataReader</a> entity allows the application to subscribe to and read data ) . . . . .	20
<a href="#">DataReaderListener</a> (The <a href="#">DataReaderListener</a> provides asynchronous notification of <a href="#">DataReader</a> events ) . . . . .	26
<a href="#">DataReaderQos</a> (Structure that holds <a href="#">DataReader</a> Quality of Service policies ) . . . . .	28
<a href="#">DataWriter</a> (The <a href="#">DataWriter</a> entity provides an interface for the application to publish (write) data. The <a href="#">DataWriter</a> is an abstract class that is extended to support a particular data type required by the application. A <a href="#">DataReader</a> is associated with, and writes on, a single <a href="#">Topic</a> ) . . . . .	31
<a href="#">DataWriterListener</a> (The <a href="#">DataWriterListener</a> provides asynchronous notification of <a href="#">DataWriter</a> events ) . . . . .	35
<a href="#">DataWriterQos</a> (Structure that holds <a href="#">DataWriter</a> Quality of Service policies ) . . . . .	36
<a href="#">DDS</a> (The 'DDS' class includes several convient constants ) . . . . .	39
<a href="#">DDS_MultiTopic</a> ( <a href="#">DDS_MultiTopic</a> provides a topic that may include data from multiple Topics )	??
<a href="#">DeadlineQosPolicy</a> (This QoS policy establishes a minimum update period for data instances ) . .	??
<a href="#">DestinationOrderQosPolicy</a> (This QoS policy controls how each <a href="#">Subscriber</a> orders received data samples ) . . . . .	??
<a href="#">DomainEntity</a> (Base class for all DDS Domain Entities ) . . . . .	45
<a href="#">DomainParticipant</a> (The <a href="#">DomainParticipant</a> is used to configure, create and destroy <a href="#">Publisher</a> , <a href="#">Subscriber</a> and <a href="#">Topic</a> objects ) . . . . .	46
<a href="#">DomainParticipantFactory</a> ( <a href="#">DomainParticipantFactory</a> constructs DomainParticipants. The ) . . .	56

DomainParticipantFactoryQos (Structure that holds DomainParticipantFactory Quality of Service policies) . . . . .	59
DomainParticipantListener (The DomainParticipantListener provides asynchronous notification of DomainParticipant events) . . . . .	60
DomainParticipantQos (Structure that holds DomainParticipant Quality of Service policies) . . . . .	62
DurabilityQosPolicy . . . . .	??
DurabilityServiceQosPolicy . . . . .	??
Duration_t (Duration_t is used to indicate a duration of time) . . . . .	??
Entity (Base class for all DDS Entities) . . . . .	63
EntityFactoryQosPolicy . . . . .	??
GroupDataQosPolicy (Allows the application to attach arbitrary information to a Publisher or Subscriber) . . . . .	??
GuardCondition (A GuardCondition is a Condition where the trigger_value is under application control) . . . . .	65
HistoryQosPolicy (Controls the amount of historical data maintained by a DataReader or DataWriter) . . . . .	??
InconsistentTopicStatus (Status related to the on_inconsistent_topic listener methods of the TopicListener structure) . . . . .	66
LatencyBudgetQosPolicy (Specifies allowable latency) . . . . .	??
LifespanQosPolicy (Specifies the maximum duration of validity of the data written by the DataWriter) . . . . .	??
LivelinessChangedStatus (Status related to the on_liveliness_changed listener methods of the DataReader, Subscriber, and DomainParticipant structures) . . . . .	67
LivelinessLostStatus (Status related to the on_liveliness_lost listener methods of the DataWriter, Publisher, and DomainParticipant structures) . . . . .	68
LivelinessQosPolicy (Determines the mechanism and parameters used by the application to determine whether an Entity is alive) . . . . .	??
OfferedDeadlineMissedStatus (Status related to the on_offered_deadline_missed listener methods of the DataWriter, Publisher, and DomainParticipant structures) . . . . .	69
OfferedIncompatibleQosStatus (Status related to the on_offered_incompatible_qos listener methods of the DataWriter, Publisher, and DomainParticipant structures) . . . . .	70
OwnershipQosPolicy (Determines instance ownership in the case of multiple writers. CoreDX DDS supports both SHARED_OWNERSHIP_QOS and EXCLUSIVE_OWNERSHIP_QOS) . . . . .	??
OwnershipStrengthQosPolicy (Defines the strength, or priority, of a Writer. The strength is used to determine ownership in the case of EXCLUSIVE_OWNERSHIP_QOS. When multiple writers publish data about the same instance, the stronger writer is considered the owner, and data from other writers is not delivered to the reader) . . . . .	??
PartitionQosPolicy . . . . .	??
PresentationQosPolicy (Controls the presentation of received data samples to the application. CoreDX DDS currently supports only the access_scope = INSTANCE_PRESENTATION_QOS policy) . . . . .	??
PublicationMatchedException (Status related to the on_publication_matched listener methods of the DataWriter, Publisher, and DomainParticipant structures) . . . . .	71
Publisher (The Publisher configures, creates, manages and destroys DataWriters) . . . . .	72
PublisherListener (The PublisherListener provides asynchronous notification of Publisher events) . . . . .	77
PublisherQos (Structure that holds Publisher Quality of Service policies) . . . . .	78



<a href="#">QueryCondition</a> (The <b>trigger_value</b> is driven by the data available, after applying the filter, in the associated <a href="#">DataReader</a> ) . . . . .	80
<a href="#">ReadCondition</a> (A <a href="#">ReadCondition</a> is a specialized <a href="#">Condition</a> associated with a <a href="#">DataReader</a> ) . . . . .	82
<a href="#">ReaderDataLifecycleQosPolicy</a> (Specifies the lifecycle behavior of data instances managed by the <a href="#">DataReader</a> ) . . . . .	??
<a href="#">ReliabilityQosPolicy</a> (Indicates the level of reliability offered/provided by the <a href="#">Entity</a> . If kind is <code>RELIABLE_RELIABILITY_QOS</code> , then the middleware will attempt to deliver all samples in the history cache. If samples are not received, then they will be retried ) . . . . .	??
<a href="#">RequestedDeadlineMissedStatus</a> (Status related to the <code>on_requested_deadline_missed</code> listener methods of the <a href="#">DataReader</a> , <a href="#">Subscriber</a> , and <a href="#">DomainParticipant</a> structures ) . . . . .	84
<a href="#">RequestedIncompatibleQosStatus</a> (Status related to the <code>on_requested_incompatible_qos</code> listener methods of the <a href="#">DataReader</a> , <a href="#">Subscriber</a> , and <a href="#">DomainParticipant</a> structures ) . . . . .	85
<a href="#">ResourceLimitsQosPolicy</a> (Specifies the resources that the Service can use to maintain data samples and instances ) . . . . .	??
<a href="#">RTPSReaderQosPolicy</a> . . . . .	??
<a href="#">RTPSWriterQosPolicy</a> . . . . .	??
<a href="#">SampleInfo</a> (The <a href="#">SampleInfo</a> structure contains information associated with each Sample. The <a href="#">DataReader.read()</a> and <a href="#">take()</a> operations return two vectors. One vector contains <a href="#">Sample(s)</a> and the other contains <a href="#">SampleInfo(s)</a> . There is a one-to-one correspondence between items in these two vectors. Each Sample is described by the corresponding <a href="#">SampleInfo</a> instance ) . . . . .	86
<a href="#">SampleLostStatus</a> (Status related to the <code>on_sample_lost</code> listener methods of the <a href="#">DataReader</a> , <a href="#">Subscriber</a> , and <a href="#">DomainParticipant</a> structures ) . . . . .	89
<a href="#">SampleRejectedStatus</a> (Status related to the <code>on_sample_rejected</code> listener methods of the <a href="#">DataReader</a> , <a href="#">Subscriber</a> , and <a href="#">DomainParticipant</a> structures ) . . . . .	90
<a href="#">StatusCondition</a> (A <a href="#">StatusCondition</a> is a condition associated with an <a href="#">Entity</a> . The <b>trigger_value</b> is driven by the communication status of the associated <a href="#">Entity</a> ) . . . . .	91
<a href="#">Subscriber</a> (The <a href="#">Subscriber</a> configures, creates, manages and destroys <a href="#">DataReaders</a> ) . . . . .	93
<a href="#">SubscriberListener</a> (The <a href="#">SubscriberListener</a> provides asynchronous notification of <a href="#">Subscriber</a> events ) . . . . .	98
<a href="#">SubscriberQos</a> (Structure that holds <code>DDS_Subscriber</code> Quality of Service policies ) . . . . .	100
<a href="#">SubscriptionMatchedStatus</a> (Status related to the <code>on_subscription_matched</code> listener methods of the <a href="#">DataReader</a> , <a href="#">Subscriber</a> , and <a href="#">DomainParticipant</a> structures ) . . . . .	102
<a href="#">Time_t</a> ( <a href="#">Time_t</a> is used to specify an point in time ) . . . . .	??
<a href="#">TimeBasedFilterQosPolicy</a> (Defines a filter based on time between samples. The <a href="#">DataReader</a> indicates that it wants at most one sample for each instance every <code>minimum_separation</code> interval ) . . . . .	??
<a href="#">Topic</a> ( <a href="#">Topic</a> is the basic description of data to be published or subscribed. A topic is identified by a <b>name</b> and a <b>type</b> . A <a href="#">Topic</a> is created by calling <a href="#">DomainParticipant.create_topic()</a> . Prior to creating a <a href="#">Topic</a> , the associated data type must be registered with the <a href="#">DomainParticipant</a> via a call to the <a href="#">TypeSupportXYZ.register_type()</a> function. [The <a href="#">register_type()</a> function is auto-generated 'type-specific' code.] ) . . . . .	103
<a href="#">TopicDataQosPolicy</a> (Allows the application to attach arbitrary information to a <a href="#">Topic</a> QoS ) . . . . .	??
<a href="#">TopicDescription</a> ( <a href="#">TopicDescription</a> is an interface that provides the foundation for <a href="#">Topic</a> , <a href="#">ContentFilteredTopic</a> , and <a href="#">MultiTopic</a> ) . . . . .	106
<a href="#">TopicListener</a> (The <a href="#">TopicListener</a> provides asynchronous notification of <a href="#">Topic</a> events ) . . . . .	107

---

<a href="#">TopicQos</a> (Structure that holds DDS_Topic Quality of Service policies) . . . . .	108
<a href="#">TransportPriorityQosPolicy</a> (A hint to the middleware to help configure the transport priority mechanism) . . . . .	??
<a href="#">UserDataQosPolicy</a> (Allows the application to attach arbitrary information to a <a href="#">DomainParticipant</a> , <a href="#">DataWriter</a> or <a href="#">DataReader</a> ) . . . . .	??
<a href="#">WaitSet</a> (A DDS_WaitSet maintains a set of <a href="#">Condition</a> objects and allows the application to wait until one or more of them have a <b>trigger_value</b> of TRUE) . . . . .	110
<a href="#">WriterDataLifecycleQosPolicy</a> (Specifies the lifecycle behavior of data instances managed by the <a href="#">DataWriter</a> . If <code>autodispose_unregistered_instances</code> is true, then the <a href="#">DataWriter</a> will automatically dispose any instances that are unregistered. <b>Note:</b> When a <a href="#">DataWriter</a> is deleted, it will automatically unregister all of its instances. With this policy == true, then all instances will also be disposed) . . . . .	??

## Chapter 5

# Not Yet Supported

Member **DataReader::create\_querycondition**(uint sample\_states, uint view\_states, uint instance\_states, String query\_e) QueryConditions are not yet supported as triggers for a WaitSet.

Class **DDS\_MultiTopic** CoreDX does not yet implement MultiTopics.

Member **DomainParticipant::create\_multitopic**(String name, String type\_name, String subscription\_expression, List< >) This is currently unsupported in the C# language binding.

Member **DomainParticipant::delete\_multitopic**(MultiTopic a\_multitopic) This is currently unsupported in the C# language binding.

Member **DomainParticipant::get\_discovered\_topic\_data**(TopicBuiltinTopicData topic\_data, InstanceHandle\_t topic\_handle) This is currently unsupported in the C# language binding.

Member **DomainParticipant::get\_discovered\_topics**(List< InstanceHandle\_t > topic\_handles) This is currently unsupported in the C# language binding.

Member **DomainParticipant::ignore\_participant**(InstanceHandle\_t handle) This is currently unsupported in the C# language binding.

Member **DomainParticipant::ignore\_publication**(InstanceHandle\_t handle) This is currently unsupported in the C# language binding.

**Member `DomainParticipant::ignore_subscription(InstanceHandle_t handle)`** This is currently unsupported in the C# language binding.

**Member `DomainParticipant::ignore_topic(InstanceHandle_t handle)`** This is currently unsupported in the C# language binding.

**Member `Publisher::begin_coherent_changes()`** This operation is not yet implemented.

**Member `Publisher::end_coherent_changes()`** This operation is not yet implemented.

**Member `Publisher::resume_publications()`** This operation is not yet implemented.

**Member `Publisher::suspend_publications()`** This operation is not yet implemented.

**Class `QueryCondition`** CoreDX DDS does not yet support QueryConditions as triggers for a WaitSet.

**Member `Subscriber::begin_access()`** This is currently unsupported in the C# language binding.

**Member `Subscriber::end_access()`** This is currently unsupported in the C# language binding.

**Member `Subscriber::get_datareaders(List< DataReader > readers, long sample_states, long view_states, long instance_states)`**  
This is currently unsupported in the C# language binding.

# Index

- absolute\_generation\_rank
  - com::toc::coredx::DDS::SampleInfo, 86
- ALIVE\_INSTANCE\_STATE
  - com::toc::coredx::DDS::DDS, 43
- ALL\_STATUS
  - com::toc::coredx::DDS::DDS, 43
- ANY\_INSTANCE\_STATE
  - com::toc::coredx::DDS::DDS, 43
- ANY\_SAMPLE\_STATE
  - com::toc::coredx::DDS::DDS, 43
- ANY\_VIEW\_STATE
  - com::toc::coredx::DDS::DDS, 43
- assert\_liveliness
  - com::toc::coredx::DDS::DataWriter, 32
  - com::toc::coredx::DDS::DomainParticipant, 47
- attach\_condition
  - com::toc::coredx::DDS::WaitSet, 110
- begin\_access
  - com::toc::coredx::DDS::Subscriber, 94
- begin\_coherent\_changes
  - com::toc::coredx::DDS::Publisher, 73
- com::toc::coredx::DDS::Condition, 15
  - get\_trigger\_value, 16
- com::toc::coredx::DDS::ContentFilteredTopic, 17
  - get\_expression\_parameters, 18
  - get\_related\_topic, 18
  - set\_expression\_parameters, 18
- com::toc::coredx::DDS::DataReader, 20
  - create\_querycondition, 21
  - create\_readcondition, 21
  - delete\_contained\_entities, 22
  - delete\_readcondition, 22
  - enable, 22
  - get\_instance\_handle, 22
  - get\_listener, 23
  - get\_liveliness\_changed\_status, 23
  - get\_matched\_publications, 23
  - get\_qos, 23
  - get\_requested\_deadline\_missed\_status, 23
  - get\_requested\_incompatible\_qos\_status, 23
  - get\_sample\_lost\_status, 24
  - get\_sample\_rejected\_status, 24
  - get\_subscriber, 24
  - get\_subscription\_matched\_status, 24
  - get\_topicdescription, 24
  - set\_listener, 24
  - set\_qos, 24
  - wait\_for\_historical\_data, 25
- com::toc::coredx::DDS::DataReaderListener, 26
  - on\_data\_available, 26
  - on\_liveliness\_changed, 26
  - on\_requested\_deadline\_missed, 26
  - on\_requested\_incompatible\_qos, 26
  - on\_sample\_lost, 26
  - on\_sample\_rejected, 26
  - on\_subscription\_matched, 26
- com::toc::coredx::DDS::DataReaderQos, 28
  - deadline, 28
  - destination\_order, 28
  - durability, 28
  - history, 29
  - latency\_budget, 29
  - liveliness, 29
  - ownership, 29
  - reader\_data\_lifecycle, 29
  - reliability, 29
  - resource\_limits, 29
  - time\_based\_filter, 29
  - user\_data, 29
- com::toc::coredx::DDS::DataWriter, 31
  - assert\_liveliness, 32
  - enable, 32

- get\_instance\_handle, 32
- get\_listener, 32
- get\_liveliness\_lost\_status, 32
- get\_matched\_subscription\_data, 32
- get\_matched\_subscriptions, 33
- get\_offered\_deadline\_missed\_status, 33
- get\_offered\_incompatible\_qos\_status, 33
- get\_publication\_matched\_status, 33
- get\_publisher, 33
- get\_qos, 33
- get\_topic, 34
- set\_listener, 34
- set\_qos, 34
- wait\_for\_acknowledgments, 34
- com::toc::coredx::DDS::DataWriterListener, 35
  - on\_liveliness\_lost, 35
  - on\_offered\_deadline\_missed, 35
  - on\_offered\_incompatible\_qos, 35
  - on\_publication\_matched, 35
- com::toc::coredx::DDS::DataWriterQos, 36
  - deadline, 36
  - destination\_order, 36
  - durability, 36
  - durability\_service, 37
  - history, 37
  - latency\_budget, 37
  - lifespan, 37
  - liveliness, 37
  - ownership, 37
  - ownership\_strength, 37
  - reliability, 37
  - resource\_limits, 37
  - transport\_priority, 37
  - user\_data, 38
  - writer\_data\_lifecycle, 38
- com::toc::coredx::DDS::DDS, 39
  - ALIVE\_INSTANCE\_STATE, 43
  - ALL\_STATUS, 43
  - ANY\_INSTANCE\_STATE, 43
  - ANY\_SAMPLE\_STATE, 43
  - ANY\_VIEW\_STATE, 43
  - DATA\_AVAILABLE\_STATUS, 43
  - DATA\_ON\_READERS\_STATUS, 43
  - DATAREADER\_QOS\_DEFAULT, 43
  - DATAWRITER\_QOS\_DEFAULT, 43
  - DEADLINE\_QOS\_POLICY\_ID, 43
  - DESTINATIONORDER\_QOS\_POLICY\_ID, 43
  - DURABILITY\_QOS\_POLICY\_ID, 43
  - DURABILITYSERVICE\_QOS\_POLICY\_ID, 43
  - DURATION\_INFINITE\_NSEC, 43
  - DURATION\_INFINITE\_SEC, 43
  - DURATION\_ZERO\_NSEC, 43
  - DURATION\_ZERO\_SEC, 43
  - ENTITYFACTORY\_QOS\_POLICY\_ID, 43
  - GROUPDATA\_QOS\_POLICY\_ID, 43
  - HANDLE\_NIL, 43
  - HISTORY\_QOS\_POLICY\_ID, 43
  - INCONSISTENT\_TOPIC\_STATUS, 43
  - LATENCYBUDGET\_QOS\_POLICY\_ID, 43
  - LENGTH\_UNLIMITED, 43
  - LIFESPAN\_QOS\_POLICY\_ID, 43
  - LIVELINESS\_CHANGED\_STATUS, 43
  - LIVELINESS\_LOST\_STATUS, 43
  - LIVELINESS\_QOS\_POLICY\_ID, 43
  - NEW\_VIEW\_STATE, 43
  - NOT\_ALIVE\_DISPOSED\_INSTANCE\_STATE, 43
  - NOT\_ALIVE\_INSTANCE\_STATE, 43
  - NOT\_ALIVE\_NO\_WRITERS\_INSTANCE\_STATE, 43
  - NOT\_NEW\_VIEW\_STATE, 43
  - NOT\_READ\_SAMPLE\_STATE, 43
  - OFFERED\_DEADLINE\_MISSED\_STATUS, 43
  - OFFERED\_INCOMPATIBLE\_QOS\_STATUS, 43
  - OWNERSHIP\_QOS\_POLICY\_ID, 43
  - OWNERSHIPSTRENGTH\_QOS\_POLICY\_ID, 43
  - PARTICIPANT\_QOS\_DEFAULT, 43
  - PARTITION\_QOS\_POLICY\_ID, 43
  - PRESENTATION\_QOS\_POLICY\_ID, 43
  - PUBLICATION\_MATCHED\_STATUS, 43
  - PUBLISHER\_QOS\_DEFAULT, 43
  - READ\_SAMPLE\_STATE, 43
  - READERDATALIFECYCLE\_QOS\_POLICY\_ID, 43
  - RELIABILITY\_QOS\_POLICY\_ID, 43
  - REQUESTED\_DEADLINE\_MISSED\_STATUS, 43

- REQUESTED\_INCOMPATIBLE\_QOS\_-  
STATUS, 43
- RESOURCELIMITS\_QOS\_POLICY\_ID, 43
- RETCODE\_ALREADY\_DELETED, 43
- RETCODE\_BAD\_PARAMETER, 43
- RETCODE\_ERROR, 43
- RETCODE\_IMMUTABLE\_POLICY, 43
- RETCODE\_INCONSISTENT\_POLICY, 43
- RETCODE\_NO\_DATA, 43
- RETCODE\_NOT\_ENABLED, 43
- RETCODE\_OK, 43
- RETCODE\_OUT\_OF\_RESOURCES, 43
- RETCODE\_PRECONDITION\_NOT\_MET, 43
- RETCODE\_TIMEOUT, 43
- RETCODE\_UNSUPPORTED, 43
- SAMPLE\_LOST\_STATUS, 43
- SAMPLE\_REJECTED\_STATUS, 43
- SUBSCRIBER\_QOS\_DEFAULT, 43
- SUBSCRIPTION\_MATCHED\_STATUS, 43
- TIMEBASEDFILTER\_QOS\_POLICY\_ID, 43
- TIMESTAMP\_INVALID\_NSEC, 43
- TIMESTAMP\_INVALID\_SEC, 43
- TOPIC\_QOS\_DEFAULT, 43
- TOPICDATA\_QOS\_POLICY\_ID, 43
- TRANSPORTPRIORITY\_QOS\_POLICY\_ID,  
43
- USERDATA\_QOS\_POLICY\_ID, 43
- WRITERDATALIFECYCLE\_QOS\_-  
POLICY\_ID, 43
- com::toc::coredx::DDS::DomainEntity, 45
- com::toc::coredx::DDS::DomainParticipant, 46
  - assert\_liveliness, 47
  - contains\_entity, 47
  - create\_contentfilteredtopic, 47
  - create\_multitopic, 48
  - create\_publisher, 48
  - create\_subscriber, 48
  - create\_topic, 48
  - delete\_contained\_entities, 49
  - delete\_contentfilteredtopic, 49
  - delete\_multitopic, 49
  - delete\_publisher, 49
  - delete\_subscriber, 50
  - delete\_topic, 50
  - enable, 50
  - find\_topic, 51
  - get\_builtin\_subscriber, 51
  - get\_current\_time, 51
  - get\_default\_publisher\_qos, 51
  - get\_default\_subscriber\_qos, 51
  - get\_default\_topic\_qos, 52
  - get\_discovered\_participant\_data, 52
  - get\_discovered\_participants, 52
  - get\_discovered\_topic\_data, 52
  - get\_discovered\_topics, 52
  - get\_domain\_id, 53
  - get\_instance\_handle, 53
  - get\_listener, 53
  - get\_qos, 53
  - ignore\_participant, 53
  - ignore\_publication, 53
  - ignore\_subscription, 54
  - ignore\_topic, 54
  - lookup\_topicdescription, 54
  - set\_default\_publisher\_qos, 54
  - set\_default\_subscriber\_qos, 55
  - set\_default\_topic\_qos, 55
  - set\_listener, 55
  - set\_qos, 55
- com::toc::coredx::DDS::DomainParticipantFactory, 56
  - create\_participant, 57
  - delete\_participant, 57
  - get\_default\_participant\_qos, 57
  - get\_instance, 57
  - Instance, 58
  - lookup\_participant, 58
  - set\_default\_participant\_qos, 58
  - set\_qos, 58
- com::toc::coredx::DDS::DomainParticipantFactoryQos, 59
  - entity\_factory, 59
- com::toc::coredx::DDS::DomainParticipantListener, 60
  - on\_data\_available, 61
  - on\_data\_on\_readers, 61
  - on\_inconsistent\_topic, 61
  - on\_liveliness\_changed, 61
  - on\_liveliness\_lost, 61
  - on\_offered\_deadline\_missed, 61
  - on\_offered\_incompatible\_qos, 61
  - on\_publication\_matched, 61

- on\_requested\_deadline\_missed, 61
- on\_requested\_incompatible\_qos, 61
- on\_sample\_lost, 61
- on\_sample\_rejected, 61
- on\_subscription\_matched, 61
- com::toc::coredx::DDS::DomainParticipantQos, 62
  - entity\_factory, 62
  - user\_data, 62
- com::toc::coredx::DDS::Entity, 63
  - enable, 63
  - get\_instance\_handle, 63
  - get\_status\_changes, 63
  - get\_statuscondition, 63
- com::toc::coredx::DDS::GuardCondition, 65
  - GuardCondition, 65
- com::toc::coredx::DDS::InconsistentTopicStatus, 66
- com::toc::coredx::DDS::LivelinessChangedStatus, 67
- com::toc::coredx::DDS::LivelinessLostStatus, 68
- com::toc::coredx::DDS::OfferedDeadlineMissedStatus, 69
- com::toc::coredx::DDS::OfferedIncompatibleQosStatus, 70
- com::toc::coredx::DDS::PublicationMatchedStatus, 71
- com::toc::coredx::DDS::Publisher, 72
  - begin\_coherent\_changes, 73
  - copy\_from\_topic\_qos, 73
  - create\_datawriter, 73
  - delete\_contained\_entities, 73
  - delete\_datawriter, 73
  - enable, 73
  - end\_coherent\_changes, 74
  - get\_default\_datawriter\_qos, 74
  - get\_instance\_handle, 74
  - get\_listener, 74
  - get\_participant, 74
  - get\_qos, 74
  - lookup\_datawriter, 75
  - resume\_publications, 75
  - set\_default\_datawriter\_qos, 75
  - set\_listener, 75
  - set\_qos, 75
  - suspend\_publications, 75
  - wait\_for\_acknowledgments, 76
- com::toc::coredx::DDS::PublisherListener, 77
  - on\_liveliness\_lost, 77
  - on\_offered\_deadline\_missed, 77
  - on\_offered\_incompatible\_qos, 77
  - on\_publication\_matched, 77
- com::toc::coredx::DDS::PublisherQos, 78
  - entity\_factory, 78
  - group\_data, 78
  - partition, 78
  - presentation, 78
- com::toc::coredx::DDS::QueryCondition, 80
  - get\_query\_expression, 81
  - get\_query\_parameters, 81
  - set\_query\_parameters, 81
- com::toc::coredx::DDS::ReadCondition, 82
  - get\_datareader, 82
  - get\_instance\_state\_mask, 82
  - get\_sample\_state\_mask, 82
  - get\_view\_state\_mask, 83
- com::toc::coredx::DDS::RequestedDeadlineMissedStatus, 84
- com::toc::coredx::DDS::RequestedIncompatibleQosStatus, 85
- com::toc::coredx::DDS::SampleInfo, 86
  - absolute\_generation\_rank, 86
  - disposed\_generation\_count, 86
  - generation\_rank, 86
  - instance\_handle, 87
  - instance\_state, 87
  - no\_writers\_generation\_count, 87
  - publication\_handle, 87
  - reception\_timestamp, 87
  - sample\_rank, 87
  - sample\_state, 87
  - source\_timestamp, 88
  - valid\_data, 88
  - view\_state, 88
- com::toc::coredx::DDS::SampleLostStatus, 89
- com::toc::coredx::DDS::SampleRejectedStatus, 90
- com::toc::coredx::DDS::StatusCondition, 91
  - get\_enabled\_statuses, 91
  - get\_entity, 91
  - set\_enabled\_statuses, 91
- com::toc::coredx::DDS::Subscriber, 93
  - begin\_access, 94
  - copy\_from\_topic\_qos, 94
  - create\_datareader, 94



- delete\_contained\_entities, 94
- delete\_datareader, 94
- enable, 95
- end\_access, 95
- get\_datareaders, 95
- get\_default\_datareader\_qos, 96
- get\_instance\_handle, 96
- get\_listener, 96
- get\_participant, 96
- get\_qos, 96
- lookup\_datareader, 96
- set\_default\_datareader\_qos, 97
- set\_listener, 97
- set\_qos, 97
- com::toc::coredx::DDS::SubscriberListener, 98
  - on\_data\_available, 99
  - on\_data\_on\_readers, 99
  - on\_liveliness\_changed, 99
  - on\_requested\_deadline\_missed, 99
  - on\_requested\_incompatible\_qos, 99
  - on\_sample\_lost, 99
  - on\_sample\_rejected, 99
  - on\_subscription\_matched, 99
- com::toc::coredx::DDS::SubscriberQos, 100
  - entity\_factory, 100
  - group\_data, 100
  - partition, 100
  - presentation, 100
- com::toc::coredx::DDS::SubscriptionMatchedStatus, 102
- com::toc::coredx::DDS::Topic, 103
  - enable, 104
  - get\_inconsistent\_topic\_status, 104
  - get\_instance\_handle, 104
  - get\_listener, 104
  - get\_qos, 104
  - set\_listener, 104
  - set\_qos, 105
- com::toc::coredx::DDS::TopicDescription, 106
- com::toc::coredx::DDS::TopicListener, 107
  - on\_inconsistent\_topic, 107
- com::toc::coredx::DDS::TopicQos, 108
  - deadline, 109
  - destination\_order, 109
  - durability, 109
  - durability\_service, 109
  - history, 109
  - latency\_budget, 109
  - lifespan, 109
  - liveliness, 109
  - ownership, 109
  - reliability, 109
  - resource\_limits, 109
  - topic\_data, 109
  - transport\_priority, 109
- com::toc::coredx::DDS::WaitSet, 110
  - attach\_condition, 110
  - destroy, 110
  - detach\_condition, 110
  - get\_conditions, 111
  - wait, 111
  - WaitSet, 110
- contains\_entity
  - com::toc::coredx::DDS::DomainParticipant, 47
- copy\_from\_topic\_qos
  - com::toc::coredx::DDS::Publisher, 73
  - com::toc::coredx::DDS::Subscriber, 94
- create\_contentfilteredtopic
  - com::toc::coredx::DDS::DomainParticipant, 47
- create\_datareader
  - com::toc::coredx::DDS::Subscriber, 94
- create\_datawriter
  - com::toc::coredx::DDS::Publisher, 73
- create\_multitopic
  - com::toc::coredx::DDS::DomainParticipant, 48
- create\_participant
  - com::toc::coredx::DDS::DomainParticipantFactory, 57
- create\_publisher
  - com::toc::coredx::DDS::DomainParticipant, 48
- create\_querycondition
  - com::toc::coredx::DDS::DataReader, 21
- create\_readcondition
  - com::toc::coredx::DDS::DataReader, 21
- create\_subscriber
  - com::toc::coredx::DDS::DomainParticipant, 48
- create\_topic
  - com::toc::coredx::DDS::DomainParticipant, 48
- DATA\_AVAILABLE\_STATUS
  - com::toc::coredx::DDS::DDS, 43
- DATA\_ON\_READERS\_STATUS

- com::toc::coredx::DDS::DDS, 43
- DATAREADER\_QOS\_DEFAULT
  - com::toc::coredx::DDS::DDS, 43
- DATAWRITER\_QOS\_DEFAULT
  - com::toc::coredx::DDS::DDS, 43
- DDS Conditions, 10
- DDS Conditions, Listeners, and WaitSets, 8
- DDS Entities, 5
- DDS Listeners, 9
- DDS Quality of Service, 7
- DDS Status Structures, 12
- DDS WaitSets, 11
- deadline
  - com::toc::coredx::DDS::DataReaderQos, 28
  - com::toc::coredx::DDS::DataWriterQos, 36
  - com::toc::coredx::DDS::TopicQos, 109
- DEADLINE\_QOS\_POLICY\_ID
  - com::toc::coredx::DDS::DDS, 43
- delete\_contained\_entities
  - com::toc::coredx::DDS::DataReader, 22
  - com::toc::coredx::DDS::DomainParticipant, 49
  - com::toc::coredx::DDS::Publisher, 73
  - com::toc::coredx::DDS::Subscriber, 94
- delete\_contentfilteredtopic
  - com::toc::coredx::DDS::DomainParticipant, 49
- delete\_datareader
  - com::toc::coredx::DDS::Subscriber, 94
- delete\_datawriter
  - com::toc::coredx::DDS::Publisher, 73
- delete\_multitopic
  - com::toc::coredx::DDS::DomainParticipant, 49
- delete\_participant
  - com::toc::coredx::DDS::DomainParticipantFactory, 57
- delete\_publisher
  - com::toc::coredx::DDS::DomainParticipant, 49
- delete\_readcondition
  - com::toc::coredx::DDS::DataReader, 22
- delete\_subscriber
  - com::toc::coredx::DDS::DomainParticipant, 50
- delete\_topic
  - com::toc::coredx::DDS::DomainParticipant, 50
- destination\_order
  - com::toc::coredx::DDS::DataReaderQos, 28
  - com::toc::coredx::DDS::DataWriterQos, 36
  - com::toc::coredx::DDS::TopicQos, 109
- DESTINATIONORDER\_QOS\_POLICY\_ID
  - com::toc::coredx::DDS::DDS, 43
- destroy
  - com::toc::coredx::DDS::WaitSet, 110
- detach\_condition
  - com::toc::coredx::DDS::WaitSet, 110
- disposed\_generation\_count
  - com::toc::coredx::DDS::SampleInfo, 86
- durability
  - com::toc::coredx::DDS::DataReaderQos, 28
  - com::toc::coredx::DDS::DataWriterQos, 36
  - com::toc::coredx::DDS::TopicQos, 109
- DURABILITY\_QOS\_POLICY\_ID
  - com::toc::coredx::DDS::DDS, 43
- durability\_service
  - com::toc::coredx::DDS::DataWriterQos, 37
  - com::toc::coredx::DDS::TopicQos, 109
- DURABILITYSERVICE\_QOS\_POLICY\_ID
  - com::toc::coredx::DDS::DDS, 43
- DURATION\_INFINITE\_NSEC
  - com::toc::coredx::DDS::DDS, 43
- DURATION\_INFINITE\_SEC
  - com::toc::coredx::DDS::DDS, 43
- DURATION\_ZERO\_NSEC
  - com::toc::coredx::DDS::DDS, 43
- DURATION\_ZERO\_SEC
  - com::toc::coredx::DDS::DDS, 43
- enable
  - com::toc::coredx::DDS::DataReader, 22
  - com::toc::coredx::DDS::DataWriter, 32
  - com::toc::coredx::DDS::DomainParticipant, 50
  - com::toc::coredx::DDS::Entity, 63
  - com::toc::coredx::DDS::Publisher, 73
  - com::toc::coredx::DDS::Subscriber, 95
  - com::toc::coredx::DDS::Topic, 104
- end\_access
  - com::toc::coredx::DDS::Subscriber, 95
- end\_coherent\_changes
  - com::toc::coredx::DDS::Publisher, 74
- entity\_factory
  - com::toc::coredx::DDS::DomainParticipantFactoryQos, 59
  - com::toc::coredx::DDS::DomainParticipantQos, 62
  - com::toc::coredx::DDS::PublisherQos, 78

- com::toc::coredx::DDS::SubscriberQos, 100
- ENTITYFACTORY\_QOS\_POLICY\_ID
  - com::toc::coredx::DDS::DDS, 43
- find\_topic
  - com::toc::coredx::DDS::DomainParticipant, 51
- generation\_rank
  - com::toc::coredx::DDS::SampleInfo, 86
- get\_builtin\_subscriber
  - com::toc::coredx::DDS::DomainParticipant, 51
- get\_conditions
  - com::toc::coredx::DDS::WaitSet, 111
- get\_current\_time
  - com::toc::coredx::DDS::DomainParticipant, 51
- get\_datareader
  - com::toc::coredx::DDS::ReadCondition, 82
- get\_datareaders
  - com::toc::coredx::DDS::Subscriber, 95
- get\_default\_datareader\_qos
  - com::toc::coredx::DDS::Subscriber, 96
- get\_default\_datawriter\_qos
  - com::toc::coredx::DDS::Publisher, 74
- get\_default\_participant\_qos
  - com::toc::coredx::DDS::DomainParticipantFactory, 57
- get\_default\_publisher\_qos
  - com::toc::coredx::DDS::DomainParticipant, 51
- get\_default\_subscriber\_qos
  - com::toc::coredx::DDS::DomainParticipant, 51
- get\_default\_topic\_qos
  - com::toc::coredx::DDS::DomainParticipant, 52
- get\_discovered\_participant\_data
  - com::toc::coredx::DDS::DomainParticipant, 52
- get\_discovered\_participants
  - com::toc::coredx::DDS::DomainParticipant, 52
- get\_discovered\_topic\_data
  - com::toc::coredx::DDS::DomainParticipant, 52
- get\_discovered\_topics
  - com::toc::coredx::DDS::DomainParticipant, 52
- get\_domain\_id
  - com::toc::coredx::DDS::DomainParticipant, 53
- get\_enabled\_statuses
  - com::toc::coredx::DDS::StatusCondition, 91
- get\_entity
  - com::toc::coredx::DDS::StatusCondition, 91
- get\_expression\_parameters
  - com::toc::coredx::DDS::ContentFilteredTopic, 18
- get\_inconsistent\_topic\_status
  - com::toc::coredx::DDS::Topic, 104
- get\_instance
  - com::toc::coredx::DDS::DomainParticipantFactory, 57
- get\_instance\_handle
  - com::toc::coredx::DDS::DataReader, 22
  - com::toc::coredx::DDS::DataWriter, 32
  - com::toc::coredx::DDS::DomainParticipant, 53
  - com::toc::coredx::DDS::Entity, 63
  - com::toc::coredx::DDS::Publisher, 74
  - com::toc::coredx::DDS::Subscriber, 96
  - com::toc::coredx::DDS::Topic, 104
- get\_instance\_state\_mask
  - com::toc::coredx::DDS::ReadCondition, 82
- get\_listener
  - com::toc::coredx::DDS::DataReader, 23
  - com::toc::coredx::DDS::DataWriter, 32
  - com::toc::coredx::DDS::DomainParticipant, 53
  - com::toc::coredx::DDS::Publisher, 74
  - com::toc::coredx::DDS::Subscriber, 96
  - com::toc::coredx::DDS::Topic, 104
- get\_liveliness\_changed\_status
  - com::toc::coredx::DDS::DataReader, 23
- get\_liveliness\_lost\_status
  - com::toc::coredx::DDS::DataWriter, 32
- get\_matched\_publications
  - com::toc::coredx::DDS::DataReader, 23
- get\_matched\_subscription\_data
  - com::toc::coredx::DDS::DataWriter, 32
- get\_matched\_subscriptions
  - com::toc::coredx::DDS::DataWriter, 33
- get\_offered\_deadline\_missed\_status
  - com::toc::coredx::DDS::DataWriter, 33
- get\_offered\_incompatible\_qos\_status
  - com::toc::coredx::DDS::DataWriter, 33
- get\_participant
  - com::toc::coredx::DDS::Publisher, 74
  - com::toc::coredx::DDS::Subscriber, 96
- get\_publication\_matched\_status
  - com::toc::coredx::DDS::DataWriter, 33
- get\_publisher
  - com::toc::coredx::DDS::DataWriter, 33

- get\_qos
  - com::toc::coredx::DDS::DataReader, 23
  - com::toc::coredx::DDS::DataWriter, 33
  - com::toc::coredx::DDS::DomainParticipant, 53
  - com::toc::coredx::DDS::Publisher, 74
  - com::toc::coredx::DDS::Subscriber, 96
  - com::toc::coredx::DDS::Topic, 104
- get\_query\_expression
  - com::toc::coredx::DDS::QueryCondition, 81
- get\_query\_parameters
  - com::toc::coredx::DDS::QueryCondition, 81
- get\_related\_topic
  - com::toc::coredx::DDS::ContentFilteredTopic, 18
- get\_requested\_deadline\_missed\_status
  - com::toc::coredx::DDS::DataReader, 23
- get\_requested\_incompatible\_qos\_status
  - com::toc::coredx::DDS::DataReader, 23
- get\_sample\_lost\_status
  - com::toc::coredx::DDS::DataReader, 24
- get\_sample\_rejected\_status
  - com::toc::coredx::DDS::DataReader, 24
- get\_sample\_state\_mask
  - com::toc::coredx::DDS::ReadCondition, 82
- get\_status\_changes
  - com::toc::coredx::DDS::Entity, 63
- get\_statuscondition
  - com::toc::coredx::DDS::Entity, 63
- get\_subscriber
  - com::toc::coredx::DDS::DataReader, 24
- get\_subscription\_matched\_status
  - com::toc::coredx::DDS::DataReader, 24
- get\_topic
  - com::toc::coredx::DDS::DataWriter, 34
- get\_topicdescription
  - com::toc::coredx::DDS::DataReader, 24
- get\_trigger\_value
  - com::toc::coredx::DDS::Condition, 16
- get\_view\_state\_mask
  - com::toc::coredx::DDS::ReadCondition, 83
- group\_data
  - com::toc::coredx::DDS::PublisherQos, 78
  - com::toc::coredx::DDS::SubscriberQos, 100
- GROUPDATA\_QOS\_POLICY\_ID
  - com::toc::coredx::DDS::DDS, 43
- GuardCondition
  - com::toc::coredx::DDS::GuardCondition, 65
- HANDLE\_NIL
  - com::toc::coredx::DDS::DDS, 43
- history
  - com::toc::coredx::DDS::DataReaderQos, 29
  - com::toc::coredx::DDS::DataWriterQos, 37
  - com::toc::coredx::DDS::TopicQos, 109
- HISTORY\_QOS\_POLICY\_ID
  - com::toc::coredx::DDS::DDS, 43
- ignore\_participant
  - com::toc::coredx::DDS::DomainParticipant, 53
- ignore\_publication
  - com::toc::coredx::DDS::DomainParticipant, 53
- ignore\_subscription
  - com::toc::coredx::DDS::DomainParticipant, 54
- ignore\_topic
  - com::toc::coredx::DDS::DomainParticipant, 54
- INCONSISTENT\_TOPIC\_STATUS
  - com::toc::coredx::DDS::DDS, 43
- Instance
  - com::toc::coredx::DDS::DomainParticipantFactory, 58
- instance\_handle
  - com::toc::coredx::DDS::SampleInfo, 87
- instance\_state
  - com::toc::coredx::DDS::SampleInfo, 87
- latency\_budget
  - com::toc::coredx::DDS::DataReaderQos, 29
  - com::toc::coredx::DDS::DataWriterQos, 37
  - com::toc::coredx::DDS::TopicQos, 109
- LATENCYBUDGET\_QOS\_POLICY\_ID
  - com::toc::coredx::DDS::DDS, 43
- LENGTH\_UNLIMITED
  - com::toc::coredx::DDS::DDS, 43
- lifespan
  - com::toc::coredx::DDS::DataWriterQos, 37
  - com::toc::coredx::DDS::TopicQos, 109
- LIFESPAN\_QOS\_POLICY\_ID
  - com::toc::coredx::DDS::DDS, 43
- liveliness
  - com::toc::coredx::DDS::DataReaderQos, 29
  - com::toc::coredx::DDS::DataWriterQos, 37
  - com::toc::coredx::DDS::TopicQos, 109

- LIVELINESS\_CHANGED\_STATUS
  - com::toc::coredx::DDS::DDS, 43
- LIVELINESS\_LOST\_STATUS
  - com::toc::coredx::DDS::DDS, 43
- LIVELINESS\_QOS\_POLICY\_ID
  - com::toc::coredx::DDS::DDS, 43
- lookup\_datareader
  - com::toc::coredx::DDS::Subscriber, 96
- lookup\_datawriter
  - com::toc::coredx::DDS::Publisher, 75
- lookup\_participant
  - com::toc::coredx::DDS::DomainParticipantFactory, 58
- lookup\_topicdescription
  - com::toc::coredx::DDS::DomainParticipant, 54
- NEW\_VIEW\_STATE
  - com::toc::coredx::DDS::DDS, 43
- no\_writers\_generation\_count
  - com::toc::coredx::DDS::SampleInfo, 87
- NOT\_REJECTED
  - status, 13
- NOT\_ALIVE\_DISPOSED\_INSTANCE\_STATE
  - com::toc::coredx::DDS::DDS, 43
- NOT\_ALIVE\_INSTANCE\_STATE
  - com::toc::coredx::DDS::DDS, 43
- NOT\_ALIVE\_NO\_WRITERS\_INSTANCE\_STATE
  - com::toc::coredx::DDS::DDS, 43
- NOT\_NEW\_VIEW\_STATE
  - com::toc::coredx::DDS::DDS, 43
- NOT\_READ\_SAMPLE\_STATE
  - com::toc::coredx::DDS::DDS, 43
- OFFERED\_DEADLINE\_MISSED\_STATUS
  - com::toc::coredx::DDS::DDS, 43
- OFFERED\_INCOMPATIBLE\_QOS\_STATUS
  - com::toc::coredx::DDS::DDS, 43
- on\_data\_available
  - com::toc::coredx::DDS::DataReaderListener, 26
  - com::toc::coredx::DDS::DomainParticipantListener, 61
  - com::toc::coredx::DDS::SubscriberListener, 99
- on\_data\_on\_readers
  - com::toc::coredx::DDS::DomainParticipantListener, 61
- com::toc::coredx::DDS::SubscriberListener, 99
- on\_inconsistent\_topic
  - com::toc::coredx::DDS::DomainParticipantListener, 61
  - com::toc::coredx::DDS::TopicListener, 107
- on\_liveliness\_changed
  - com::toc::coredx::DDS::DataReaderListener, 26
  - com::toc::coredx::DDS::DomainParticipantListener, 61
  - com::toc::coredx::DDS::SubscriberListener, 99
- on\_liveliness\_lost
  - com::toc::coredx::DDS::DataWriterListener, 35
  - com::toc::coredx::DDS::DomainParticipantListener, 61
  - com::toc::coredx::DDS::PublisherListener, 77
- on\_offered\_deadline\_missed
  - com::toc::coredx::DDS::DataWriterListener, 35
  - com::toc::coredx::DDS::DomainParticipantListener, 61
  - com::toc::coredx::DDS::PublisherListener, 77
- on\_offered\_incompatible\_qos
  - com::toc::coredx::DDS::DataWriterListener, 35
  - com::toc::coredx::DDS::DomainParticipantListener, 61
  - com::toc::coredx::DDS::PublisherListener, 77
- on\_publication\_matched
  - com::toc::coredx::DDS::DataWriterListener, 35
  - com::toc::coredx::DDS::DomainParticipantListener, 61
  - com::toc::coredx::DDS::PublisherListener, 77
- on\_requested\_deadline\_missed
  - com::toc::coredx::DDS::DataReaderListener, 26
  - com::toc::coredx::DDS::DomainParticipantListener, 61
  - com::toc::coredx::DDS::SubscriberListener, 99
- on\_requested\_incompatible\_qos
  - com::toc::coredx::DDS::DataReaderListener, 26
  - com::toc::coredx::DDS::DomainParticipantListener, 61
  - com::toc::coredx::DDS::SubscriberListener, 99
- on\_sample\_lost
  - com::toc::coredx::DDS::DataReaderListener, 26

- com::toc::coredx::DDS::DomainParticipantListener, 61
- com::toc::coredx::DDS::SubscriberListener, 99
- on\_sample\_rejected
  - com::toc::coredx::DDS::DataReaderListener, 26
  - com::toc::coredx::DDS::DomainParticipantListener, 61
  - com::toc::coredx::DDS::SubscriberListener, 99
- on\_subscription\_matched
  - com::toc::coredx::DDS::DataReaderListener, 26
  - com::toc::coredx::DDS::DomainParticipantListener, 61
  - com::toc::coredx::DDS::SubscriberListener, 99
- ownership
  - com::toc::coredx::DDS::DataReaderQos, 29
  - com::toc::coredx::DDS::DataWriterQos, 37
  - com::toc::coredx::DDS::TopicQos, 109
- OWNERSHIP\_QOS\_POLICY\_ID
  - com::toc::coredx::DDS::DDS, 43
- ownership\_strength
  - com::toc::coredx::DDS::DataWriterQos, 37
- OWNERSHIPSTRENGTH\_QOS\_POLICY\_ID
  - com::toc::coredx::DDS::DDS, 43
- PARTICIPANT\_QOS\_DEFAULT
  - com::toc::coredx::DDS::DDS, 43
- partition
  - com::toc::coredx::DDS::PublisherQos, 78
  - com::toc::coredx::DDS::SubscriberQos, 100
- PARTITION\_QOS\_POLICY\_ID
  - com::toc::coredx::DDS::DDS, 43
- presentation
  - com::toc::coredx::DDS::PublisherQos, 78
  - com::toc::coredx::DDS::SubscriberQos, 100
- PRESENTATION\_QOS\_POLICY\_ID
  - com::toc::coredx::DDS::DDS, 43
- publication\_handle
  - com::toc::coredx::DDS::SampleInfo, 87
- PUBLICATION\_MATCHED\_STATUS
  - com::toc::coredx::DDS::DDS, 43
- PUBLISHER\_QOS\_DEFAULT
  - com::toc::coredx::DDS::DDS, 43
- READ\_SAMPLE\_STATE
  - com::toc::coredx::DDS::DDS, 43
  - reader\_data\_lifecycle
    - com::toc::coredx::DDS::DataReaderQos, 29
  - READERDATALIFECYCLE\_QOS\_POLICY\_ID
    - com::toc::coredx::DDS::DDS, 43
  - reception\_timestamp
    - com::toc::coredx::DDS::SampleInfo, 87
  - REJECTED\_BY\_INSTANCE\_LIMIT
    - status, 13
  - REJECTED\_BY\_SAMPLES\_LIMIT
    - status, 13
  - REJECTED\_BY\_SAMPLES\_PER\_INSTANCE\_LIMIT
    - status, 13
  - reliability
    - com::toc::coredx::DDS::DataReaderQos, 29
    - com::toc::coredx::DDS::DataWriterQos, 37
    - com::toc::coredx::DDS::TopicQos, 109
  - RELIABILITY\_QOS\_POLICY\_ID
    - com::toc::coredx::DDS::DDS, 43
  - REQUESTED\_DEADLINE\_MISSED\_STATUS
    - com::toc::coredx::DDS::DDS, 43
  - REQUESTED\_INCOMPATIBLE\_QOS\_STATUS
    - com::toc::coredx::DDS::DDS, 43
  - resource\_limits
    - com::toc::coredx::DDS::DataReaderQos, 29
    - com::toc::coredx::DDS::DataWriterQos, 37
    - com::toc::coredx::DDS::TopicQos, 109
  - RESOURCELIMITS\_QOS\_POLICY\_ID
    - com::toc::coredx::DDS::DDS, 43
  - resume\_publications
    - com::toc::coredx::DDS::Publisher, 75
  - RETCODE\_ALREADY\_DELETED
    - com::toc::coredx::DDS::DDS, 43
  - RETCODE\_BAD\_PARAMETER
    - com::toc::coredx::DDS::DDS, 43
  - RETCODE\_ERROR
    - com::toc::coredx::DDS::DDS, 43
  - RETCODE\_IMMUTABLE\_POLICY
    - com::toc::coredx::DDS::DDS, 43
  - RETCODE\_INCONSISTENT\_POLICY
    - com::toc::coredx::DDS::DDS, 43
  - RETCODE\_NO\_DATA
    - com::toc::coredx::DDS::DDS, 43
  - RETCODE\_NOT\_ENABLED
    - com::toc::coredx::DDS::DDS, 43

- RETCODE\_OK
  - com::toc::coredx::DDS::DDS, 43
- RETCODE\_OUT\_OF\_RESOURCES
  - com::toc::coredx::DDS::DDS, 43
- RETCODE\_PRECONDITION\_NOT\_MET
  - com::toc::coredx::DDS::DDS, 43
- RETCODE\_TIMEOUT
  - com::toc::coredx::DDS::DDS, 43
- RETCODE\_UNSUPPORTED
  - com::toc::coredx::DDS::DDS, 43
- SAMPLE\_LOST\_STATUS
  - com::toc::coredx::DDS::DDS, 43
- sample\_rank
  - com::toc::coredx::DDS::SampleInfo, 87
- SAMPLE\_REJECTED\_STATUS
  - com::toc::coredx::DDS::DDS, 43
- sample\_state
  - com::toc::coredx::DDS::SampleInfo, 87
- SampleRejectedStatusKind
  - status, 13
- set\_default\_datareader\_qos
  - com::toc::coredx::DDS::Subscriber, 97
- set\_default\_datawriter\_qos
  - com::toc::coredx::DDS::Publisher, 75
- set\_default\_participant\_qos
  - com::toc::coredx::DDS::DomainParticipantFactory, 58
- set\_default\_publisher\_qos
  - com::toc::coredx::DDS::DomainParticipant, 54
- set\_default\_subscriber\_qos
  - com::toc::coredx::DDS::DomainParticipant, 55
- set\_default\_topic\_qos
  - com::toc::coredx::DDS::DomainParticipant, 55
- set\_enabled\_statuses
  - com::toc::coredx::DDS::StatusCondition, 91
- set\_expression\_parameters
  - com::toc::coredx::DDS::ContentFilteredTopic, 18
- set\_listener
  - com::toc::coredx::DDS::DataReader, 24
  - com::toc::coredx::DDS::DataWriter, 34
  - com::toc::coredx::DDS::DomainParticipant, 55
  - com::toc::coredx::DDS::Publisher, 75
  - com::toc::coredx::DDS::Subscriber, 97
  - com::toc::coredx::DDS::Topic, 104
- set\_qos
  - com::toc::coredx::DDS::DataReader, 24
  - com::toc::coredx::DDS::DataWriter, 34
  - com::toc::coredx::DDS::DomainParticipant, 55
  - com::toc::coredx::DDS::DomainParticipantFactory, 58
  - com::toc::coredx::DDS::Publisher, 75
  - com::toc::coredx::DDS::Subscriber, 97
  - com::toc::coredx::DDS::Topic, 105
- set\_query\_parameters
  - com::toc::coredx::DDS::QueryCondition, 81
- source\_timestamp
  - com::toc::coredx::DDS::SampleInfo, 88
- status
  - NOT\_REJECTED, 13
  - REJECTED\_BY\_INSTANCE\_LIMIT, 13
  - REJECTED\_BY\_SAMPLES\_LIMIT, 13
  - REJECTED\_BY\_SAMPLES\_PER\_INSTANCE\_LIMIT, 13
  - SampleRejectedStatusKind, 13
- SUBSCRIBER\_QOS\_DEFAULT
  - com::toc::coredx::DDS::DDS, 43
- SUBSCRIPTION\_MATCHED\_STATUS
  - com::toc::coredx::DDS::DDS, 43
- suspend\_publications
  - com::toc::coredx::DDS::Publisher, 75
- time\_based\_filter
  - com::toc::coredx::DDS::DataReaderQos, 29
- TIMEBASEDFILTER\_QOS\_POLICY\_ID
  - com::toc::coredx::DDS::DDS, 43
- TIMESTAMP\_INVALID\_NSEC
  - com::toc::coredx::DDS::DDS, 43
- TIMESTAMP\_INVALID\_SEC
  - com::toc::coredx::DDS::DDS, 43
- topic\_data
  - com::toc::coredx::DDS::TopicQos, 109
- TOPIC\_QOS\_DEFAULT
  - com::toc::coredx::DDS::DDS, 43
- TOPICDATA\_QOS\_POLICY\_ID
  - com::toc::coredx::DDS::DDS, 43
- transport\_priority
  - com::toc::coredx::DDS::DataWriterQos, 37
  - com::toc::coredx::DDS::TopicQos, 109
- TRANSPORTPRIORITY\_QOS\_POLICY\_ID
  - com::toc::coredx::DDS::DDS, 43

---

user\_data  
    com::toc::coredx::DDS::DataReaderQos, 29  
    com::toc::coredx::DDS::DataWriterQos, 38  
    com::toc::coredx::DDS::DomainParticipantQos,  
        62

USERDATA\_QOS\_POLICY\_ID  
    com::toc::coredx::DDS::DDS, 43

valid\_data  
    com::toc::coredx::DDS::SampleInfo, 88

view\_state  
    com::toc::coredx::DDS::SampleInfo, 88

wait  
    com::toc::coredx::DDS::WaitSet, 111

wait\_for\_acknowledgments  
    com::toc::coredx::DDS::DataWriter, 34  
    com::toc::coredx::DDS::Publisher, 76

wait\_for\_historical\_data  
    com::toc::coredx::DDS::DataReader, 25

WaitSet  
    com::toc::coredx::DDS::WaitSet, 110

writer\_data\_lifecycle  
    com::toc::coredx::DDS::DataWriterQos, 38

WRITERDATALIFECYCLE\_QOS\_POLICY\_ID  
    com::toc::coredx::DDS::DDS, 43