CoreDX ™ Data Distribution Service

The leading Small Footprint DDS Middleware

Java Reference Manual

Twin Oaks Computing, Inc

Castle Rock, CO 80108

Nov 2011

Trademarks

Copy and Use Restrictions

DISCLAIMER OF WARRANTY

Contact

Twin Oaks Computing, Inc
755 Maleta Ln, Ste 203
Castle Rock, CO 80108
(720) 733-7906
contact@twinoakscomputing.com
http://www.twinoakscomputing.com
http://twitter.com/CoreDX_DDS

Contents

1 Overview **1**

2 Data Structure Documentation **5**

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

3 API Documentation **15**

3.1 Condition Class Reference 15

3.2 ContentFilteredTopic Class Reference 17

3.3 DataReader Class Reference 19

3.4 DataReaderListener Interface Reference 25

3.5 DataReaderQos Class Reference 27

3.6 DataWriter Class Reference 30

3.7 DataWriterListener Interface Reference 34

3.8 DataWriterQos Class Reference 36

3.9 DDS Class Reference 39

3.10 DeadlineQosPolicy Class Reference 40

3.11 DestinationOrderQosPolicy Class Reference 41

3.12 DomainEntity Class Reference 42
3.13 DomainId_t Class Reference 43
3.14 DomainParticipant Class Reference 44
3.15 DomainParticipantFactory Class Reference 54
3.16 DomainParticipantFactoryQos Class Reference 57
3.17 DomainParticipantListener Interface Reference 58
3.18 DomainParticipantQos Class Reference 62
3.19 DurabilityQosPolicy Class Reference 63
3.20 DurabilityServiceQosPolicy Class Reference 64
3.21 Duration_t Class Reference 65
3.22 Entity Class Reference 66
3.23 EntityFactoryQosPolicy Class Reference 68
3.24 EntityNameQosPolicy Class Reference 69
3.25 FooDataReader Class Reference 70
3.26 FooDataWriter Class Reference 78
3.27 FooTypeSupport Class Reference 81
3.28 GroupDataQosPolicy Class Reference 83
3.29 GuardCondition Class Reference 84
3.30 HistoryQosPolicy Class Reference 85
3.31 InconsistentTopicStatus Class Reference 86
3.32 InstanceHandle_t Class Reference 87
3.33 LatencyBudgetQosPolicy Class Reference 88
3.34 LifespanQosPolicy Class Reference 89
3.35 LivelinessChangedStatus Class Reference 90
3.36 LivelinessLostStatus Class Reference 92
3.37 LivelinessQosPolicy Class Reference 93
3.38 MultiTopic Class Reference 94
3.39 OfferedDeadlineMissedStatus Class Reference 96
3.40 OfferedIncompatibleQosStatus Class Reference 97
3.41 OwnershipQosPolicy Class Reference 98
3.42 OwnershipStrengthQosPolicy Class Reference 99

3.43 ParticipantBuiltinTopicDataDataReader Class Reference 100
3.44 PartitionQosPolicy Class Reference 101
3.45 PresentationQosPolicy Class Reference 102
3.46 PublicationBuiltinTopicDataDataReader Class Reference 103
3.47 PublicationMatchedStatus Class Reference 104
3.48 Publisher Class Reference 106
3.49 PublisherListener Interface Reference 111
3.50 PublisherQos Class Reference 113
3.51 QueryCondition Class Reference 115
3.52 ReadCondition Class Reference 117
3.53 ReaderDataLifecycleQosPolicy Class Reference 119
3.54 ReliabilityQosPolicy Class Reference 120
3.55 RequestedDeadlineMissedStatus Class Reference 121
3.56 RequestedIncompatibleQosStatus Class Reference 122
3.57 ResourceLimitsQosPolicy Class Reference 123
3.58 SampleInfo Class Reference 124
3.59 SampleLostStatus Class Reference 127
3.60 SampleRejectedStatus Class Reference 128
3.61 StatusCondition Class Reference 129
3.62 Subscriber Class Reference 131
3.63 SubscriberListener Interface Reference 136
3.64 SubscriberQos Class Reference 139
3.65 SubscriptionBuiltinTopicDataDataReader Class Reference 141
3.66 SubscriptionMatchedStatus Class Reference 142
3.67 Time_t Class Reference 144
3.68 TimeBasedFilterQosPolicy Class Reference 145
3.69 Topic Class Reference 146
3.70 TopicDataQosPolicy Class Reference 149
3.71 TopicDescription Interface Reference 150
3.72 TopicListener Interface Reference 152
3.73 TopicQos Class Reference 153

3.74 TransportPriorityQosPolicy Class Reference 156
3.75 TypeSupport Interface Reference 157
3.76 UserDataQosPolicy Class Reference 159
3.77 WaitSet Class Reference 160
3.78 WriterDataLifecycleQosPolicy Class Reference 162

4 Data Structure Index **163**
4.1 Class List 163

5 Not Yet Supported **169**

Chapter 1

Overview

Welcome to the CoreDX DDS for Java 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 Java 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 "Java" 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 Java 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
    - com.toc.coredx.DDS.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 ContentFilteredTopic

  ContentFilteredTopic provides a topic that may include data filtered from a related Topic.

- 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 DomainEntity

  Base class for all DDS Domain Entities.

- class DomainParticipant

  The DomainParticipant is used to configure, create and destroy Publisher, Subscriber and Topic objects.

- class DomainParticipantFactory

  DomainParticipantFactory constructs DomainParticipants. The.

- class Entity

  Base class for all DDS Entities.

- class FooDataReader

  The FooDataReader is an example specialized DataReader (generated by the coredx_ddl compiler) for reading data samples of type 'Foo'.

- class FooDataWriter

  The FooDataWriter is an example specialized DataWriter (generated by the coredx_ddl compiler) for writing data samples of type 'Foo'.

- class FooTypeSupport

  The FooTypeSupport class impliments the TypeSupport inteface for the Data Type 'Foo'.

- class MultiTopic

  MultiTopic provides a topic that may include data from multiple Topics.

- 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.].*

2.2 DDS Quality of Service

Classes

- class DataReaderQos

  Structure that holds DataReader Quality of Service policies.

- class DataWriterQos

  Structure that holds DataWriter Quality of Service policies.

- class DomainParticipantFactoryQos

  Structure that holds DomainParticipantFactory Quality of Service policies.

- class DomainParticipantQos

  Structure that holds DomainParticipant 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 TopicQos

  Structure that holds DDS_Topic Quality of Service policies.

2.3 DDS Conditions, Listeners, and WaitSets

Modules

- DDS Listeners
- DDS Conditions
- DDS WaitSets

2.4 DDS Listeners

Classes

- interface DataReaderListener

  The DataReaderListener provides asynchronous notification of DataReader events. This listener can be installed during DataReader creation, DomainParticipant_create_yyy(), as well as by calling DataReader_set_listener().

- interface DataWriterListener

  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().

- interface DomainParticipantListener
- interface PublisherListener
- interface SubscriberListener

  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().

- interface TopicListener

  The DDS_TopicListener provides asynchronous notification of DDS_Topic events. This listener can be installed during Topic creation (DDS_DomainParticipant_create_topic() and related) as well as by calling DDS_Topic_set_listener().

2.5 DDS Conditions

Classes

- class Condition

  A Condition can be added to a DDS_WaitSet to provide synchronous event notification.

- class GuardCondition

  *A GuardCondition is a Condition where the **trigger_value** is under application control.*

- class QueryCondition

  *The **trigger_value** is driven by the data available, after applying the filter, in the associated DataReader.*

- class ReadCondition

  A ReadCondition is a specialized Condition associated with a 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

  InconsistentTopicStatus provides status related to the on_inconsistent_topic listener methods of the TopicListener structure.

- class LivelinessChangedStatus

  Status related to the on_liveliness_changed listener methods of the DataReader, Subscriber, and DomainParticipant structures.

- class LivelinessLostStatus

  Status related to the on_liveliness_lost listener methods of the DataWriter, Publisher, and DomainParticipant structures.

- 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 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 SampleLostStatus

  Status related to the on_sample_lost 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 SubscriptionMatchedStatus

  Status related to the on_subscription_matched listener methods of the DataReader, Subscriber, and DomainParticipant structures.

2.7.1 Detailed Description

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

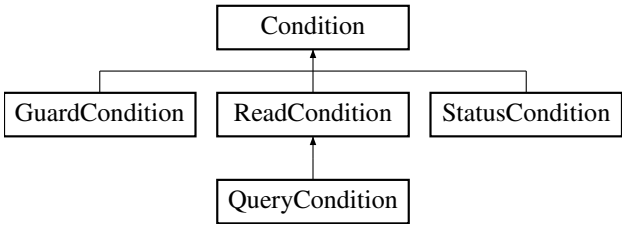
Chapter 3

API Documentation

3.1 Condition Class Reference

A Condition can be added to a DDS_WaitSet to provide synchronous event notification.

Inheritance diagram for Condition:



Public Member Functions

- boolean get_trigger_value ()

3.1.1 Detailed Description

A Condition can be added to a DDS_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 boolean 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.

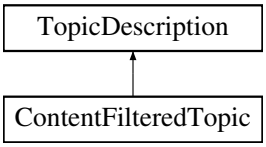
The documentation for this class was generated from the following file:

- /home/ctucker/coredx_v3.4rc/src/dds_java/com/toc/coredx/DDS/Condition.java

3.2 ContentFilteredTopic Class Reference

ContentFilteredTopic provides a topic that may include data filtered from a related Topic.

Inheritance diagram for ContentFilteredTopic:



Public Member Functions

- DomainParticipant get_participant ()
  *This operation returns the parent **DomainParticipant** of the ContentFilteredTopic.*
- String get_type_name ()
  *This operation returns **type_name** of the ContentFilteredTopic.*
- String get_name ()
  *This operation returns **topic name** of the ContentFilteredTopic.*
- Topic get_related_topic ()
  This returns the real Topic associated with the ContentFilteredTopic.
- Vector get_expression_parameters ()
  This accesses the current set of parameters used by the ContentFilteredTopic.
- ReturnCode_t set_expression_parameters (Vector 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.

3.2.2 Member Function Documentation

3.2.2.1 Vector get_expression_parameters ( ) `[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 (Vector *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 refering to paramters in a filter_-expression is the percent sign ” followed by a number. The number is the index of the paramter in the **filter_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.

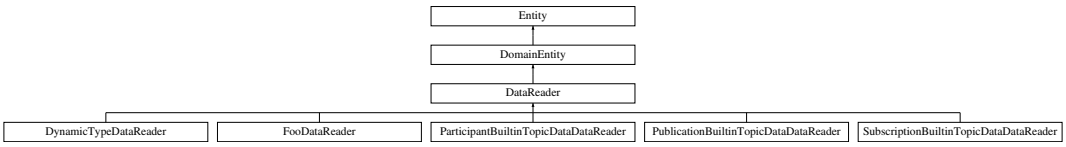
The documentation for this class was generated from the following file:

- /home/ctucker/coredx_v3.4rc/src/dds_java/com/toc/coredx/DDS/ContentFilteredTopic.java

3.3 DataReader Class Reference

The DataReader entity allows the application to subscribe to and read data.

Inheritance diagram for DataReader:



Public Member Functions

- ReturnCode_t enable ()
- ReadCondition create_readcondition (long sample_states, long view_states, long instance_states)
- QueryCondition create_querycondition (long sample_states, long view_states, long instance_states, String query_expression, Vector 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, long mask)
- DataReaderListener get_listener ()
- TopicDescription get_topicdescription ()
- Subscriber get_subscriber ()
- ReturnCode_t get_sample_rejected_status (SampleRejectedStatus status)
- ReturnCode_t get_liveliness_changed_status (LivelinessChangedStatus status)
- ReturnCode_t get_requested_deadline_missed_status (RequestedDeadlineMissedStatus status)
- ReturnCode_t get_requested_incompatible_qos_status (RequestedIncompatibleQosStatus status)
- ReturnCode_t get_subscription_matched_status (SubscriptionMatchedStatus status)
- ReturnCode_t get_sample_lost_status (SampleLostStatus status)
- ReturnCode_t wait_for_historical_data (Duration_t max_wait)
- ReturnCode_t get_matched_publications (InstanceHandleSeq publication_handles)
- ReturnCode_t get_matched_publication_data (PublicationBuiltinTopicData publication_data, InstanceHandle_t publication_handle)

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).

See also

com.toc.coredx.DDS.FooDataReader

3.3.2 Member Function Documentation

3.3.2.1 QueryCondition create_querycondition (long *sample_states,* long *view_states,* long *instance_states,* String *query_expression,* Vector *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 (long *sample_states,* long *view_states,* long *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 ReturnCode_t enable ( ) [inline]

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 DataReaderListener get_listener ( ) [inline]

This operation returns the currently installed DataReaderListener.

3.3.2.7 ReturnCode_t get_liveliness_changed_status (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.

3.3.2.8 ReturnCode_t get_matched_publication_data (PublicationBuiltinTopicData *publication_data,* InstanceHandle_t *publication_handle*) [inline]

This operation returns data that describes a particular matched DataWriter identified by **publication_handle.** An appropriate handle can be obtained through a call to DataReader.get_matched_publications().

If **publication_handle** does not identify a currently matched DataWriter, this routine will return ReturnCode_t.RETCODE_PRECONDITION_NOT_MET.

3.3.2.9 ReturnCode_t get_matched_publications (InstanceHandleSeq *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 (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 (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 (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 (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 Subscribier that contains DataReader **dr.**

3.3.2.16 ReturnCode_t get_subscription_matched_status (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,* long *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_java/com/toc/coredx/DDS/DataReader.java

3.4 DataReaderListener Interface Reference

The DataReaderListener provides asynchronous notification of DataReader events. This listener can be installed during DataReader creation, DomainParticipant_create_yyy(), as well as by calling DataReader_set_listener().

Public Member Functions

- void on_requested_incompatible_qos (DataReader the_reader, RequestedIncompatibleQosStatus status)
- void on_sample_rejected (DataReader the_reader, SampleRejectedStatus status)
- void on_liveliness_changed (DataReader the_reader, LivelinessChangedStatus status)
- void on_data_available (DataReader the_reader)
- void on_subscription_matched (DataReader the_reader, SubscriptionMatchedStatus status)
- void on_sample_lost (DataReader the_reader, SampleLostStatus status)
- long get_nil_mask ()

Package Functions

- void on_requested_deadline_missed (DataReader the_reader, RequestedDeadlineMissedStatus status)

3.4.1 Detailed Description

The DataReaderListener provides asynchronous notification of DataReader events. This listener can be installed during DataReader creation, DomainParticipant_create_yyy(), as well as by calling DataReader_set_listener().

3.4.2 Member Function Documentation

3.4.2.1 long get_nil_mask ( )

get_nil_mask() returns a bitmask indicating which listener methods (if any) should be considered NIL, and therefore, should not be invoked.

3.4.2.2 void on_data_available (DataReader *the_reader*)

Called when the CoreDX infrastructure detects that new data or data state information is available.

The **status** argument provides a snapshot of the status at the time the listener was invoked.

3.4.2.3 void on_liveliness_changed (DataReader *the_reader,* LivelinessChangedStatus *status*)

Called when the CoreDX infrastructure detects that the liveliness of a matched DataWriter has changed (either 'active' or 'inactive').

The **status** argument provides a snapshot of the status at the time the listener was invoked.

3.4.2.4 void on_requested_deadline_missed (DataReader *the_reader,* RequestedDeadlineMissedStatus *status*) [package]

Called when the CoreDX infrastructure detects that the deadline specified in the DataReader QoS DEADLINE policy was not satisfied for a data instance.

The **status** argument provides a snapshot of the status at the time the listener was invoked.

3.4.2.5 void on_requested_incompatible_qos (DataReader *the_reader,* RequestedIncompatibleQosStatus *status*)

Called when the CoreDX infrastructure detects that the DataReader requested a QoS policy that was incompatible with that offered by a DataWriter.

The **status** argument provides a snapshot of the status at the time the listener was invoked.

3.4.2.6 void on_sample_lost (DataReader *the_reader,* SampleLostStatus *status*)

Called when the CoreDX infrastructure detects that a sample has been lost (never received).

The **status** argument provides a snapshot of the status at the time the listener was invoked.

3.4.2.7 void on_sample_rejected (DataReader *the_reader,* SampleRejectedStatus *status*)

Called when the CoreDX infrastructure detects that a sample has been rejected.

The **status** argument provides a snapshot of the status at the time the listener was invoked.

3.4.2.8 void on_subscription_matched (DataReader *the_reader,* SubscriptionMatchedStatus *status*)

Called when the CoreDX infrastructure detects that the DataReader has matched or ceased to be matched with a DataWriter.

The **status** argument provides a snapshot of the status at the time the listener was invoked.

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

- /home/ctucker/coredx_v3.4rc/src/dds_java/com/toc/coredx/DDS/DataReaderListener.java

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(DataReaderQos) set_qos()
DataReader::get_qos(DataReaderQos) get_qos()
Subscriber::create_datareader(TopicDescription topic, DataReaderQos qos, DataReaderListener listener, long mask) create_datareader()
Subscriber::set_default_datareader_qos(DataReaderQos) set_default_datareader_qos()
Subscriber::get_default_datareader_qos(DataReaderQos) 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.

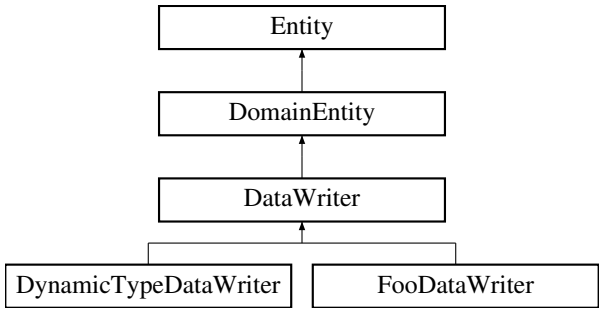
The documentation for this class was generated from the following file:

- /home/ctucker/coredx_v3.4rc/src/dds_java/com/toc/coredx/DDS/DataReaderQos.java

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

- ReturnCode_t enable ()
- ReturnCode_t set_qos (DataWriterQos qos)
- ReturnCode_t get_qos (DataWriterQos qos)
- ReturnCode_t set_listener (DataWriterListener new_listener, long 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 (LivelinessLostStatus status)
- ReturnCode_t get_offered_deadline_missed_status (OfferedDeadlineMissedStatus status)
- ReturnCode_t get_offered_incompatible_qos_status (OfferedIncompatibleQosStatus status)
- ReturnCode_t get_publication_matched_status (PublicationMatchedStatus status)
- ReturnCode_t get_matched_subscriptions (InstanceHandleSeq 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 LIVELINESS QoS setting is MANUAL_BY_PARTICIPANT_LIVELINESS_QOS or MANUAL_BY_TOPIC_-LIVELINESS_QOS; otherwise, it has no effect.

The **write** operation automatically asserts livelines on the DataWriter and its DomainParticipant. Therefore, **assert_liveliness** is required only if hte application is not writing data frequently enough to satisfy the LIVELINESS setting.

3.6.2.2 ReturnCode_t enable ( ) [inline]

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 DataWriterListener get_listener ( ) [inline]

This operation returns the currently installed DataWriterListener.

3.6.2.4 ReturnCode_t get_liveliness_lost_status (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.

3.6.2.5 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.

3.6.2.6 ReturnCode_t get_matched_subscriptions (InstanceHandleSeq *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).

3.6.2.7 ReturnCode_t get_offered_deadline_missed_status (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.

3.6.2.8 ReturnCode_t get_offered_incompatible_qos_status (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.

3.6.2.9 ReturnCode_t get_publication_matched_status (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.

3.6.2.10 Publisher get_publisher ( ) [inline]

Returns the Publisher that contains DataWriter **dw.**

3.6.2.11 ReturnCode_t get_qos (DataWriterQos *qos*) [inline]

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

3.6.2.12 Topic get_topic ( ) [inline]

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

3.6.2.13 ReturnCode_t set_listener (DataWriterListener *new_listener,* long *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.

3.6.2.14 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.

3.6.2.15 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_java/com/toc/coredx/DDS/DataWriter.java

3.7 DataWriterListener Interface Reference

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().

Public Member Functions

- void on_offered_incompatible_qos (DataWriter writer, OfferedIncompatibleQosStatus status)
- void on_liveliness_lost (DataWriter writer, LivelinessLostStatus status)
- void on_publication_matched (DataWriter writer, PublicationMatchedStatus status)
- long get_nil_mask ()

Package Functions

- void on_offered_deadline_missed (DataWriter writer, OfferedDeadlineMissedStatus status)

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().

3.7.2 Member Function Documentation

3.7.2.1 long get_nil_mask ( )

get_nil_mask() returns a bitmask indicating which listener methods (if any) should be considered NIL, and therefore, should not be invoked.

3.7.2.2 void on_liveliness_lost (DataWriter *writer,* LivelinessLostStatus *status*)

Called when the CoreDX infrastructure detects that the DataWriter has not satisfied its LIVELINESS QoS setting.

The **status** argument provides a snapshot of the status at the time the listener was invoked.

3.7.2.3 void on_offered_deadline_missed (DataWriter *writer,* OfferedDeadlineMissedStatus *status*) [package]

Called when the CoreDX infrastructure detects that the deadline that the DDS_DataWriter has offered Through the DEADLINE QoS was not satisfied. That is, the DataWriter has failed to update an instance

with the frequency specified in the DEADLINE QoS.

The **status** argument provides a snapshot of the status at the time the listener was invoked.

3.7.2.4 void on_offered_incompatible_qos (DataWriter *writer,* OfferedIncompatibleQosStatus *status*)

Called when the CoreDX infrastructure detects that the DataWriter has offered a QoS policy setting that is incompatible with that requested by a potentially matching DataReader.

The **status** argument provides a snapshot of the status at the time the listener was invoked.

3.7.2.5 void on_publication_matched (DataWriter *writer,* PublicationMatchedStatus *status*)

Called when the CoreDX infrastructure detects that the DataWriter has matched with a DataReader or has ceased to be matched with a DataReader.

The **status** argument provides a snapshot of the status at the time the listener was invoked.

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

- /home/ctucker/coredx_v3.4rc/src/dds_java/com/toc/coredx/DDS/DataWriterListener.java

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(DataWriterQos) set_qos()
DataWriter::get_qos(DataWriterQos) get_qos()
Publisher::create_datawriter(Topic topic, DataWriterQos qos, DataWriterListener listener, long mask) create_datawriter()
Publisher::set_default_datawriter_qos(DataWriterQos) set_default_datawriter_qos()
Publisher::get_default_datawriter_qos(DataWriterQos) 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_java/com/toc/coredx/DDS/DataWriterQos.java

3.9 DDS Class Reference

Static Public Member Functions

- static String qos_policy_str (int qos_policy_id)
- static String error_str (ReturnCode_t err)

3.9.1 Detailed Description

Provides various constants and utility routines used throughout CoreDX DDS.

3.9.2 Member Function Documentation

3.9.2.1 static String error_str (ReturnCode_t *err*) [inline, static]

Gets a string describing the provided 'err'.

3.9.2.2 static String qos_policy_str (int *qos_policy_id*) [inline, static]

Gets a string describing a QoS Policy ID.

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

- /home/ctucker/coredx_v3.4rc/src/dds_java/com/toc/coredx/DDS/DDS.java

3.10 DeadlineQosPolicy Class Reference

This QoS policy establishes a minimum update period for data instances.

3.10.1 Detailed Description

This QoS policy establishes a minimum update period for data instances. DataWriters specify that each data instance will be updated at least once every 'period'. A DataReader requests that the Writer commit to updating each instance at least once every 'period'.

The policy is compatible (between a Writer and a Reader) if the offered deadline period <= requested deadline period.

If a DataWriter fails to meet the update period, then the DataWriter is notified by a call to the offered_-deadline_missed listener, and the DataReader is notified by a call to the requested_deadlin_missed listener.

See also

com.toc.coredx.DDS.DataReaderListener::on_requested_deadline_missed(DataReader the_reader, RequestedDeadlineMissedStatus status) on_requested_deadline_missed
com.toc.coredx.DDS.DataWriterListener::on_offered_deadline_missed(DataWriter writer, OfferedDeadlineMissedStatus status) on_offered_deadline_missed

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

- /home/ctucker/coredx_v3.4rc/src/dds_java/com/toc/coredx/DDS/DeadlineQosPolicy.java

3.11 DestinationOrderQosPolicy Class Reference

This QoS policy controls how each Subscriber orders received data samples.

3.11.1 Detailed Description

This QoS policy controls how each Subscriber orders received data samples. Can be **BY_RECEPTION_TIMESTAMP** or **BY_SOURCE_TIMESTAMP**. The samples are ordered either based on their order of reception, or by the order of timestamps generated by the source (DataWriter).

CoreDX DDS currently implements only BY_RECEPTION_TIMESTAMP.

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

- /home/ctucker/coredx_v3.4rc/src/dds_java/com/toc/coredx/DDS/DestinationOrderQosPolicy.java

3.12 DomainEntity Class Reference

Base class for all DDS Domain Entities.

Inheritance diagram for DomainEntity:

3.12.1 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_java/com/toc/coredx/DDS/DomainEntity.java

3.13 DomainId_t Class Reference

3.13.1 Detailed Description

Identifies a DDS Domain. The DomainParticipant is a member of a Domain as identified by the DomainId provided at creation time. Domains are represented as integers.

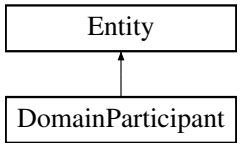
The documentation for this class was generated from the following file:

- /home/ctucker/coredx_v3.4rc/src/dds_java/com/toc/coredx/DDS/DomainId_t.java

3.14 DomainParticipant Class Reference

The DomainParticipant is used to configure, create and destroy Publisher, Subscriber and Topic objects.

Inheritance diagram for DomainParticipant:



Public Member Functions

- ReturnCode_t enable ()

  Enables the DomainParticipant.

- ReturnCode_t get_qos (DomainParticipantQos qos)
- ReturnCode_t set_qos (DomainParticipantQos qos)
- ReturnCode_t set_listener (DomainParticipantListener new_listener, long mask)
- DomainParticipantListener get_listener ()
- Publisher create_publisher (PublisherQos qos, PublisherListener listener, long mask)
- ReturnCode_t delete_publisher (Publisher p)
- Subscriber create_subscriber (SubscriberQos qos, SubscriberListener listener, long mask)
- ReturnCode_t delete_subscriber (Subscriber s)
- Topic create_topic (String topic_name, String type_name, TopicQos qos, TopicListener listener, long mask)
- ContentFilteredTopic create_contentfilteredtopic (String name, Topic related_topic, String filter_expression, Vector filter_parameters)
- ReturnCode_t delete_contentfilteredtopic (ContentFilteredTopic cft)
- MultiTopic create_multitopic (String name, String type_name, String subscription_expression, Vector expression_params)
- ReturnCode_t delete_multitopic (MultiTopic a_multitopic)
- ReturnCode_t delete_topic (Topic topic)
- Topic find_topic (String topic_name, Duration_t timeout)
- TopicDescription lookup_topicdescription (String name)
- Subscriber get_builtin_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 (InstanceHandleSeq participant_handles)
- ReturnCode_t get_discovered_participant_data (ParticipantBuiltinTopicData participant_data, InstanceHandle_t participant_handle)
- ReturnCode_t get_discovered_topics (Vector topic_handles)
- ReturnCode_t get_discovered_topic_data (TopicBuiltinTopicData topic_data, InstanceHandle_t topic_handle)
- boolean contains_entity (InstanceHandle_t handle)
- ReturnCode_t get_current_time (Time_t current_time)
- ReturnCode_t register_type (TypeSupport ts, String type_name)

  Registers a TypeSupport with the Participant.

3.14.1 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.

3.14.2 Member Function Documentation

3.14.2.1 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 on ore more DataWriter objects with LIVELINES QoS set to MANUAL_BY_PARTICIPANT_LIVELINESS_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'.

3.14.2.2 boolean 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.

3.14.2.3 ContentFilteredTopic create_contentfilteredtopic (String *name,* Topic *related_topic,* String *filter_expression,* Vector *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 paramters is the percent sign '' followed by a number. The number is the index of the paramter in the **filter_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.

3.14.2.4 MultiTopic create_multitopic (String *name,* String *type_name,* String *subscription_expression,* Vector *expression_params*) [inline]

Not Yet Supported

This is currently unsupported in the Java language binding.

3.14.2.5 Publisher create_publisher (PublisherQos *qos,* PublisherListener *listener,* long *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 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.

3.14.2.6 Subscriber create_subscriber (SubscriberQos *qos,* SubscriberListener *listener,* long *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

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.

3.14.2.7 Topic create_topic (String *topic_name,* String *type_name,* TopicQos *qos,* TopicListener *listener,* long *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().

3.14.2.8 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.

3.14.2.9 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 DomainPartipant **dp;** otherwise ReturnCode_t.RETCODE_-PRECONDITION_NOT_MET will be returned.

3.14.2.10 ReturnCode_t delete_multitopic (MultiTopic *a_multitopic*) [inline]

Not Yet Supported

This is currently unsupported in the Java language binding.

3.14.2.11 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.**

3.14.2.12 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.**

3.14.2.13 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 DomainPartipant **dp;** otherwise ReturnCode_t.RETCODE_PRECONDITION_NOT_MET will be returned.

3.14.2.14 ReturnCode_t enable ( ) [inline]

Enables the DomainParticipant.

A DomainParticipant is created either enabled or not based on the DomainParticipantFactoryQoS setting **entity_factory.** When a DomainPartcipant 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.

3.14.2.15 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 **timout** 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().

3.14.2.16 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

3.14.2.17 ReturnCode_t get_current_time (Time_t *current_time*) [inline]

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

3.14.2.18 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.

3.14.2.19 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.

3.14.2.20 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.

3.14.2.21 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.

3.14.2.22 ReturnCode_t get_discovered_participants (InstanceHandleSeq *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().

3.14.2.23 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 Java language binding.

3.14.2.24 ReturnCode_t get_discovered_topics (Vector *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 Java language binding.

3.14.2.25 long get_domain_id ( ) [inline]

Gets the **domain_id** to which the DomainParticipant belongs.

3.14.2.26 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.

3.14.2.27 ReturnCode_t get_qos (DomainParticipantQos *qos*) [inline]

Gets the DomainParticipantQoS settings of the DomainParticipant.

3.14.2.28 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 Java language binding.

3.14.2.29 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 Java language binding.

3.14.2.30 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 Java language binding.

3.14.2.31 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 Java language binding.

3.14.2.32 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.

3.14.2.33 ReturnCode_t register_type (TypeSupport *ts,* String *type_name*) [inline]

Registers a TypeSupport with the Participant.

Associates a TypeSupport object with the provided 'type_name'. This TypeSupport will be used to handle data that has a matching type_name. If a TypeSupport has already been registered for the provied type_name, and the new type does not match the previously registered type, then an error is returned and the previously registered TypeSupport is maintained. [Upon error, the new type is not registered.]

3.14.2.34 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.

3.14.2.35 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.

3.14.2.36 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.

3.14.2.37 ReturnCode_t set_listener (DomainParticipantListener *new_listener,* long *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.

3.14.2.38 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_java/com/toc/coredx/DDS/DomainParticipant.java

3.15 DomainParticipantFactory Class Reference

DomainParticipantFactory constructs DomainParticipants. The.

Public Member Functions

- DomainParticipant create_participant (long domain_id, DomainParticipantQos qos, DomainParticipantListener listener, long mask)
- ReturnCode_t delete_participant (DomainParticipant participant)
- DomainParticipant lookup_participant (long domain_id)
- ReturnCode_t set_default_participant_qos (DomainParticipantQos qos)
- ReturnCode_t get_default_participant_qos (DomainParticipantQos qos)
- ReturnCode_t set_qos (DomainParticipantFactoryQos qos)
- ReturnCode_t get_qos (DomainParticipantFactoryQos qos)

Static Public Member Functions

- static DomainParticipantFactory get_instance ()

3.15.1 Detailed Description

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

Author

Twin Oaks Computing, Inc

Since

2.2

3.15.2 Member Function Documentation

3.15.2.1 DomainParticipant create_participant (long *domain_id*, DomainParticipantQos *qos*, DomainParticipantListener *listener*, long *mask*) [inline]

Create and initialize a DomainParticipant. 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.

Parameters

domain_id The DomainId of the created DomainParticipant.

qos The QoS to use for the create DP.

listener The listener that will receive event notifications, can be null

mask A mask indicating which methods within 'listener' should be invoked.

See also

DDS::PARTICIPANT_QOS_DEFAULT
DomainParticipantListener
DDS::ALL_STATUS

3.15.2.2 ReturnCode_t delete_participant (DomainParticipant *participant*) [inline]

Destroy a DomainParticipant. This routine will fail if all Entities (Publishers, Subscribers, etc) created through this DomainParticipant have not yet been deleted. (In this case, RETCODE_PRECONDITION_NOT_MET will be returned.)

Parameters

participant A DomainParticipant previosly created by a call to create_participant()

3.15.2.3 ReturnCode_t get_default_participant_qos (DomainParticipantQos *qos*) [inline]

Provides access to the default Participant qos held in the factory. The provided **qos** argument is populated with the default qos settings.

3.15.2.4 static DomainParticipantFactory get_instance ( ) [inline, static]

Get access to the singleton DomainParticipantFactory.

3.15.2.5 ReturnCode_t get_qos (DomainParticipantFactoryQos *qos*) [inline]

Gets the current QoS settings of the DomainParticipantFactory.

3.15.2.6 DomainParticipant lookup_participant (long *domain_id*) [inline]

Find a DomainParticipant with the provided domain_id. If there are multiple participants in existence with the specified id, then one of them will be returned. (Which one is not specified.)

3.15.2.7 ReturnCode_t set_default_participant_qos (DomainParticipantQos *qos*) [inline]

Sets the default DomainParticipantQos held in the factory. This default qos will be used during subsequent calls to 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, ReturnCode_-t.RETCODE_INCONSISTENT_POLICY will be returned, and no changes will be made to the DomainParticipantFactory.

3.15.2.8 ReturnCode_t set_qos (DomainParticipantFactoryQos *qos*) [inline]

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, ReturnCode_-t.RETCODE_INCONSISTENT_POLICY will be returned, and no changes will be made to the DomainParticipantFactory.

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

- /home/ctucker/coredx_v3.4rc/src/dds_java/com/toc/coredx/DDS/DomainParticipantFactory.java

3.16 DomainParticipantFactoryQos Class Reference

Structure that holds DomainParticipantFactory Quality of Service policies.

Public Attributes

- EntityFactoryQosPolicy entity_factory

3.16.1 Detailed Description

Structure that holds DomainParticipantFactory Quality of Service policies.

See also

DomainParticipantFactory::set_qos(DomainParticipantFactoryQos) set_qos()
DomainParticipantFactory::get_qos(DomainParticipantFactoryQos) get_qos()

3.16.2 Member Data Documentation

3.16.2.1 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_java/com/toc/coredx/DDS/DomainParticipantFactoryQos.java

3.17 DomainParticipantListener Interface Reference

Public Member Functions

- void on_inconsistent_topic (Topic the_topic, InconsistentTopicStatus status)
- void on_offered_deadline_missed (DataWriter writer, OfferedDeadlineMissedStatus status)
- void on_offered_incompatible_qos (DataWriter writer, OfferedIncompatibleQosStatus status)
- void on_liveliness_lost (DataWriter writer, LivelinessLostStatus status)
- void on_publication_matched (DataWriter writer, PublicationMatchedStatus status)
- void on_requested_deadline_missed (DataReader the_reader, RequestedDeadlineMissedStatus status)
- void on_requested_incompatible_qos (DataReader the_reader, RequestedIncompatibleQosStatus status)
- void on_sample_rejected (DataReader the_reader, SampleRejectedStatus status)
- void on_liveliness_changed (DataReader the_reader, LivelinessChangedStatus status)
- void on_data_available (DataReader the_reader)
- void on_subscription_matched (DataReader the_reader, SubscriptionMatchedStatus status)
- void on_sample_lost (DataReader the_reader, SampleLostStatus status)
- void on_data_on_readers (Subscriber the_subscriber)
- long get_nil_mask ()

3.17.1 Detailed Description

The DomainParticipantListener provides asynchronous notification of DDS_DomainParticipant events. This listener can be installed during DomainParticipant creation, DomainParticipantFactory_create_participant(), as well as by calling DomainParticipant_set_listener().

3.17.2 Member Function Documentation

3.17.2.1 long get_nil_mask ( )

get_nil_mask() returns a bitmask indicating which listener methods (if any) should be considered NIL, and therefore, should not be invoked.

3.17.2.2 void on_data_available (DataReader *the_reader*)

Called when the CoreDX infrastructure detects that a DataReader, contained in any Subscriber in this DomainParticipant, has new data or data state information available. This listener is invoked only if the concerned DataReader and Subscriber do not have an **on_data_available** listener installed.

The **status** argument provides a snapshot of the status at the time the listener was invoked.

3.17.2.3 void on_data_on_readers (Subscriber *the_subscriber*)

Called when the CoreDX infrastructure detects that a DataReader, contained in any Subscriber in this DomainParticipant, has new data or data state information available. This listener is invoked only if the concerned Subscriber does not have an **on_data_on_readers** listener installed.

The **status** argument provides a snapshot of the status at the time the listener was invoked.

3.17.2.4 void on_inconsistent_topic (Topic *the_topic,* InconsistentTopicStatus *status*)

Called when the CoreDX infrastructure detects that a Topic contained within this DomainParticipant has characteristics taht are different (inconsistent) with another existing topic. This listener is invoked only if the concerned Topic does not have an **on_inconsistent_topic** listener installed.

The **status** argument provides a snapshot of the status at the time the listener was invoked.

3.17.2.5 void on_liveliness_changed (DataReader *the_reader,* LivelinessChangedStatus *status*)

Called when the CoreDX infrastructure detects that the liveliness of a DataWriter, matched to a DataReader within any Subscriber within this DomainParticipant, has changed (either 'active' or 'inactive'). This listener is invoked only if the concerned DataReader and Subscriber do not have an **on_liveliness_changed** listener installed.

The **status** argument provides a snapshot of the status at the time the listener was invoked.

3.17.2.6 void on_liveliness_lost (DataWriter *writer,* LivelinessLostStatus *status*)

Called when the CoreDX infrastructure detects that a DataWriter contained in any Publisher within this DomainParticipant has not satisfied its LIVELINESS QoS setting. This listener is invoked only if the concerned DataWriter and Publisher do not have an **on_liveliness_lost** listener installed.

The **status** argument provides a snapshot of the status at the time the listener was invoked.

3.17.2.7 void on_offered_deadline_missed (DataWriter *writer,* OfferedDeadlineMissedStatus *status*)

Called when the CoreDX infrastructure detects that a DataWriter contained in any Publisher within this DomainParticipant has failed to meet its DEADLINE QoS commitment. This listener is invoked only if the concerned DataWriter and Publisher do not have an **on_offered_deadline_missed** listener installed.

The **status** argument provides a snapshot of the status at the time the listener was invoked.

3.17.2.8 void on_offered_incompatible_qos (DataWriter *writer,* OfferedIncompatibleQosStatus *status*)

Called when the CoreDX infrastructure detects that a DataWriter contained in any Publisher within this DomainParticipant has offered a QoS policy setting that is incompatible with that requested by a potentially matching DataReader. This listener is invoked only if the concerned DataWriter and Publisher do not have an **on_offered_incompatible_qos** listener installed.

The **status** argument provides a snapshot of the status at the time the listener was invoked.

3.17.2.9 void on_publication_matched (DataWriter *writer,* PublicationMatchedStatus *status*)

Called when the CoreDX infrastructure detects that a DataWriter contained in any Publisher within this DomainParticipant has matched with a DataReader or has ceased to be matched with a DataReader. This listener is invoked only if the concerned DataWriter and Publisher do not have an **on_publication_matched** listener installed.

The **status** argument provides a snapshot of the status at the time the listener was invoked.

3.17.2.10 void on_requested_deadline_missed (DataReader *the_reader,* RequestedDeadlineMissedStatus *status*)

Called when the CoreDX infrastructure detects that the QoS DEADLINE policy of a DataReader, contained in any Subscriber of this DomainParticipant, was not satisfied for a data instance. This listener is invoked only if the concerned DataReader and Subscriber do not have an **on_requested_deadline_missed** listener installed.

The **status** argument provides a snapshot of the status at the time the listener was invoked.

3.17.2.11 void on_requested_incompatible_qos (DataReader *the_reader,* RequestedIncompatibleQosStatus *status*)

Called when the CoreDX infrastructure detects that a DataReader contained in any Subscriber within this DomainParticipant, has requested a QoS policy that was incompatible with that offered by a DataWriter. This listener is invoked only if the concerned DataReader and Subscriber do not have an **on_requested_incompatible_qos** listener installed.

The **status** argument provides a snapshot of the status at the time the listener was invoked.

3.17.2.12 void on_sample_lost (DataReader *the_reader,* SampleLostStatus *status*)

Ccalled when the CoreDX infrastructure detects that a DataReader, contained in any Subscriber in this DomainParticipant, has lost a sample (never received). This listener is invoked only if the concerned DataReader and Subscriber do not have an **on_sample_lost** listener installed.

The **status** argument provides a snapshot of the status at the time the listener was invoked.

3.17.2.13 void on_sample_rejected (DataReader *the_reader,* SampleRejectedStatus *status*)

Called when the CoreDX infrastructure detects that a DataReader contained in any Subscriber within the DomainParticipan has rejected a sample. This listener is invoked only if the concerned DataReader and Subscriber do not have an **on_sample_rejected** listener installed.

The **status** argument provides a snapshot of the status at the time the listener was invoked.

3.17.2.14 void on_subscription_matched (DataReader *the_reader,* SubscriptionMatchedStatus *status*)

Called when the CoreDX infrastructure detects that a DataReader, contained in any Subscriber in this DomainParticipant, has matched or ceased to be matched with a DataWriter. This listener is invoked only if the concerned DataReader and Subscriber do not have an **on_subscription_matched** listener installed.

The **status** argument provides a snapshot of the status at the time the listener was invoked.

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

- /home/ctucker/coredx_v3.4rc/src/dds_java/com/toc/coredx/DDS/DomainParticipantListener.java

3.18 DomainParticipantQos Class Reference

Structure that holds DomainParticipant Quality of Service policies.

Public Attributes

- UserDataQosPolicy user_data
- EntityFactoryQosPolicy entity_factory
- PeerParticipantQosPolicy peer_participants

3.18.1 Detailed Description

Structure that holds DomainParticipant Quality of Service policies.

See also

DomainParticipant::set_qos(DomainParticipantQos) set_qos()
DomainParticipant::get_qos(DomainParticipantQos) get_qos()
DomainParticipantFactory::create_participant(long domain_id, DomainParticipantQos qos, DomainParticipantListener listener, long mask) create_participant()
DomainParticipantFactory::set_default_participant_qos(DomainParticipantQos) set_default_participant_qos()
DomainParticipantFactory::get_default_participant_qos(DomainParticipantQos) get_default_participant_qos()

3.18.2 Member Data Documentation

3.18.2.1 EntityFactoryQosPolicy entity_factory

Controls the behavior of the DomainParticipant **create** operations.

3.18.2.2 PeerParticipantQosPolicy peer_participants

Specifies a list of DomainParticipant peers to attempt communication with. If empty, default Discovery is used.

3.18.2.3 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_java/com/toc/coredx/DDS/DomainParticipantQos.java

3.19 DurabilityQosPolicy Class Reference

3.19.1 Detailed Description

The DurablityQosPolicy controls the durablity of data. The DDS API identifies several degrees of data durability.

1. VOLATILE: The data is volatile, and is not persisted beyond the initial publication.
2. TRANSIENT_LOCAL: The data is persisted locally within the source DataWriter. If that DataWriter is destroyed, or the containing application exits, the data is no longer available.
3. TRANSIENT: The data is persisted beyond the lifecycle of the originating DataWriter, and is available even after that DataWriter has been destroyed. However, data does not persist a reboot of the computer.
4. PERSISTENT: The data is persisted to 'off-line' storage, and is available even after a system reboot.

A DataWriter can offer to provide a level of durablity for data that it generates, and a DataReader requests the level of durability that it requires. If the DataReader requests a 'higher' level of durability than that offered by the DataWriter, then the QoS policy is incompatible.

CoreDX DDS currently supports VOLATILE and TRANSIENT_LOCAL.

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

- /home/ctucker/coredx_v3.4rc/src/dds_java/com/toc/coredx/DDS/DurabilityQosPolicy.java

3.20 DurabilityServiceQosPolicy Class Reference

3.20.1 Detailed Description

The DurablityServiceQosPolicy supports the durablity of data. The DDS API identifies several degrees of data durability.

1. VOLATILE: The data is volatile, and is not persisted beyond the initial publication.
2. TRANSIENT_LOCAL: The data is persisted locally within the source DataWriter. If that DataWriter is destroyed, or the containing application exits, the data is no longer available.
3. TRANSIENT: The data is persisted beyond the lifecycle of the originating DataWriter, and is available even after that DataWriter has been destroyed. However, data does not persist a reboot of the computer.
4. PERSISTENT: The data is persisted to 'off-line' storage, and is available even after a system reboot.

If a durability of TRANSIENT or PERSISTENT is specified, then a service beyond the source DataWriter must be established to provide advanced data durability. This QoS policy helps to configure that durablity service.

CoreDX DDS currently supports VOLATILE and TRANSIENT_LOCAL.

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

- /home/ctucker/coredx_v3.4rc/src/dds_java/com/toc/coredx/DDS/DurabilityServiceQosPolicy.java

3.21 Duration_t Class Reference

Represents a time duration with nanosecond resolution.

3.21.1 Detailed Description

Represents a time duration with nanosecond resolution.

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

- /home/ctucker/coredx_v3.4rc/src/dds_java/com/toc/coredx/DDS/Duration_t.java

3.22 Entity Class Reference

Base class for all DDS Entities.

Inheritance diagram for Entity:

Public Member Functions

- StatusCondition get_statuscondition ()
- int get_status_changes ()
- ReturnCode_t enable ()
- InstanceHandle_t get_instance_handle ()

3.22.1 Detailed Description

Base class for all DDS Entities.

3.22.2 Member Function Documentation

3.22.2.1 ReturnCode_t enable ( ) [inline]

Enable this Entity. An Entity will begin to communicate only after it is enabled.

Reimplemented in DataReader, DataWriter, DomainParticipant, Publisher, and Subscriber.

3.22.2.2 InstanceHandle_t get_instance_handle ( ) [inline]

Gets the handle that locally identifies this Entity.

3.22.2.3 int get_status_changes ( ) [inline]

Gets the current status changes from this Entity.

3.22.2.4 StatusCondition get_statuscondition ( ) [inline]

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_java/com/toc/coredx/DDS/Entity.java

3.23 EntityFactoryQosPolicy Class Reference

3.23.1 Detailed Description

Controls the behavior of entity factories. Several DDS entities act as factories to create, control, and destroy other DDS entities. For example, the DomainParticipant is a factory for Publisher, Subscriber, and Topic entities. This QoS controls the behavior of a factory.

A DDS entity can be enabled at creation time. If the autoenable_created_entities is set to **true**, then the factory will automatically call enable() on the entity during creation. Otherwise, the entity is created but not enabled. The application must specifically call enable() on the returned entity.

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

- /home/ctucker/coredx_v3.4rc/src/dds_java/com/toc/coredx/DDS/EntityFactoryQosPolicy.java

3.24 EntityNameQosPolicy Class Reference

3.24.1 Detailed Description

Assigns a name to an Entity. This is a CoreDX DDS extension, and is provided for convenience. Entity names assigned through this QoS are not used internally by the middleware, they are provided for debug and analysis purposes.

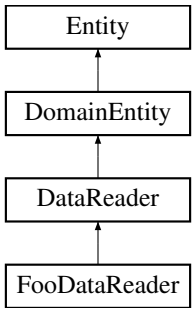
The documentation for this class was generated from the following file:

- /home/ctucker/coredx_v3.4rc/src/dds_java/com/toc/coredx/DDS/EntityNameQosPolicy.java

3.25 FooDataReader Class Reference

The FooDataReader is an example specialized DataReader (generated by the coredx_ddl compiler) for reading data samples of type 'Foo'.

Inheritance diagram for FooDataReader:



Public Member Functions

- ReturnCode_t get_key_value (Foo key_holder, InstanceHandle_t handle)
- InstanceHandle_t lookup_instance (Foo instance_data)
- ReturnCode_t return_loan (FooSeq received_data, SampleInfoSeq sample_infos)
- ReturnCode_t read (FooSeq received_data, SampleInfoSeq sample_infos, int max_samples, long sample_states, long view_states, long instance_states)
- ReturnCode_t take (FooSeq received_data, SampleInfoSeq sample_infos, int max_samples, long sample_states, long view_states, long instance_states)
- ReturnCode_t read_w_condition (FooSeq received_data, SampleInfoSeq sample_infos, int max_samples, ReadCondition condition)
- ReturnCode_t take_w_condition (FooSeq received_data, SampleInfoSeq sample_infos, int max_samples, ReadCondition condition)
- ReturnCode_t read_next_sample (Foo received_data, SampleInfo sample_info)
- ReturnCode_t take_next_sample (Foo received_data, SampleInfo sample_info)
- ReturnCode_t read_instance (FooSeq received_data, SampleInfoSeq sample_infos, int max_samples, InstanceHandle_t handle, long sample_states, long view_states, long instance_states)
- ReturnCode_t take_instance (FooSeq received_data, SampleInfoSeq sample_infos, int max_samples, InstanceHandle_t handle, long sample_states, long view_states, long instance_states)
- ReturnCode_t read_next_instance (FooSeq received_data, SampleInfoSeq sample_infos, int max_samples, InstanceHandle_t prev_handle, long sample_states, long view_states, long instance_states)
- ReturnCode_t take_next_instance (FooSeq received_data, SampleInfoSeq sample_infos, int max_samples, InstanceHandle_t prev_handle, long sample_states, long view_states, long instance_states)
- ReturnCode_t read_next_instance_w_condition (FooSeq received_data, SampleInfoSeq sample_infos, int max_samples, InstanceHandle_t handle, ReadCondition condition)

- ReturnCode_t take_next_instance_w_condition (FooSeq received_data, SampleInfoSeq sample_infos, int max_samples, InstanceHandle_t handle, ReadCondition condition)

3.25.1 Detailed Description

The FooDataReader is an example specialized DataReader (generated by the coredx_ddl compiler) for reading data samples of type 'Foo'. The FooDataReader is an implementation of the base DataReader class that has been extended to support the 'Foo' data type.

Note: the FooDataReader is included here for documentation purposes only, and does not really exist in the com.toc.coredx.DDS package.

3.25.2 Member Function Documentation

3.25.2.1 ReturnCode_t get_key_value (Foo *key_holder,* InstanceHandle_t *handle*)

This routine will populate the data structure indicated by **key_holder** with the key infomation identified by **handle.**

This routine is data type specific. The generated type specific DataReader includes an implementation of this routine which should be used to support type-safety.

3.25.2.2 InstanceHandle_t lookup_instance (Foo *instance_data*)

Returns the handle that identifies the data instance provided in **instance_data.** The 'key' field values of the data are associated with a unique handle.

This routine is data type specific. The generated type specific DataReader includes an implementation of this routine which should be used to support type-safety.

3.25.2.3 ReturnCode_t read (FooSeq *received_data,* SampleInfoSeq *sample_infos,* int *max_samples,* long *sample_states,* long *view_states,* long *instance_states*)

This operation accesses a collection of data values (with associated SampleInfo) within the DataReader. This routine, and the related **take(),** provide the interface for an application to access published data. There are several varieties of **read()** and **take(),** to facilitate different access patterns or approaches.

The primary difference between **read()** and **take()** is that **take()** removes all returned data samples from the DataReader while **read()** does not. Sequential **read()** calls will return the same data samples each time (if nothing else changes); while sequential **take()** calls will return data samples for only the first call. Subsequent **take()** calls will return an empty collection (if no new data arrives).

The specific behavior of **read()** depends on several things: input paramters, the QoS settings of the DataReader, and the state of recieved data. First, the **received_data** and **sample_infos** arguments affect the following:

- how many samples are returned, and
- whether the returned data should be 'loaned' or copied.

The argument **max_samples** is used to further limit the number of samples returned.

The **sample_states, view_states,** and **instance_states** arguments are used to selectively add data samples to the returned collections. These arguments indicate the desired 'states' for data samples and instances. These state arguments are bit masks; they can be the bit-wise OR of several individual state flags or they may use the special 'ANY' constants (e.g.: DDS.ANY_SAMPLE_STATE). Only samples that have a matching state for all three categories are added to the returned collection.

The order of samples in the returned collections is determined by the **PRESENTATION** and **DESTINATION_ORDER** QoS policies.

The returned collection is held in **recieved_data** and **samples_infos.** These two sequences operate together to represent a sequence of pairs (data, SampleInfo). Each data item in **received_data** has a corresponding entry in **sample_infos** that provides associated 'meta-data'. See SampleInfo for a description of this meta-data.

In CoreDX DDS, the returned sequences contain 'loaned' data. This provides zero-copy access to the data, and provides a very efficient data access mechanism. Becuase the data is 'loaned' to the application, the application is required to indicate when it is finished accessing the data. This is accomplished by calling return_loan().

The **read()** operation will set the SampleInfo.sample_state to DDS.READ_SAMPLE_STATE.

The **read()** operation may set the SampleInfo.view_state to DDS.NOT_NEW_VIEW_STATE, if a sample of the most recent generation of the instance is read.

If there is no data found, then the **read()** will return RETCODE_NO_DATA.

This routine is data type specific. The generated type specific DataReader includes an implementation of this routine which should be used to support type-safety.

3.25.2.4 ReturnCode_t read_instance (FooSeq *received_data,* SampleInfoSeq *sample_infos,* int *max_samples,* InstanceHandle_t *handle,* long *sample_states,* long *view_states,* long *instance_states*)

This operation accesses a collection of data values (with associated SampleInfo), belonging to a particular instance, within the DataReader.

This routine is data type specific. The generated type specific DataReader includes an implementation of this routine which should be used to support type-safety.

See also

com.toc.coredx.DDS.FooDataReader::read(FooSeq received_data, SampleInfoSeq sample_infos, int max_samples, long sample_states, long view_states, long instance_states) read()

3.25.2.5 ReturnCode_t read_next_instance (FooSeq *received_data,* SampleInfoSeq *sample_infos,* int *max_samples,* InstanceHandle_t *prev_handle,* long *sample_states,* long *view_states,* long *instance_states*)

This operation accesses a collection of data values (with associated SampleInfo), instance by instance, within the DataReader. To start,pass HANDLE_NIL for parameter prev_handle. After the last instance (in order) has been taken, this routine will return RETCODE_NO_DATA.

This routine is data type specific. The generated type specific DataReader includes an implementation of this routine which should be used to support type-safety.

See also

com.toc.coredx.DDS.FooDataReader::read(FooSeq received_data, SampleInfoSeq sample_infos, int max_samples, long sample_states, long view_states, long instance_states) read()

3.25.2.6 ReturnCode_t read_next_instance_w_condition (FooSeq *received_data,* SampleInfoSeq *sample_infos,* int *max_samples,* InstanceHandle_t *handle,* ReadCondition *condition*)

This operation accesses a collection of data values (with associated SampleInfo), instance by instance subject to a filter, within the DataReader.

This routine is data type specific. The generated type specific DataReader includes an implementation of this routine which should be used to support type-safety.

See also

com.toc.coredx.DDS.FooDataReader::read(FooSeq received_data, SampleInfoSeq sample_infos, int max_samples, long sample_states, long view_states, long instance_states) read()

3.25.2.7 ReturnCode_t read_next_sample (Foo *received_data,* SampleInfo *sample_info*)

This operation accesses a data value (with associated SampleInfo) within the DataReader.

This routine is data type specific. The generated type specific DataReader includes an implementation of this routine which should be used to support type-safety.

See also

com.toc.coredx.DDS.FooDataReader::read(FooSeq received_data, SampleInfoSeq sample_infos, int max_samples, long sample_states, long view_states, long instance_states) read()

3.25.2.8 ReturnCode_t read_w_condition (FooSeq *received_data,* SampleInfoSeq *sample_infos,* int *max_samples,* ReadCondition *condition*)

This operation accesses a collection of data values (with associated SampleInfo), subject to a filter, within the DataReader.

This routine is data type specific. The generated type specific DataReader includes an implementation of this routine which should be used to support type-safety.

See also

com.toc.coredx.DDS.FooDataReader::read(FooSeq received_data, SampleInfoSeq sample_infos, int max_samples, long sample_states, long view_states, long instance_states) read()

3.25.2.9 ReturnCode_t return_loan (FooSeq *received_data,* SampleInfoSeq *sample_infos*)

Returns data and sample_info values to a DataReader. When an application calls DataReader **read()** or **take()** operations, these routines 'loan' data and SampleInfo values to the application. [This is an optimization that avoids extra copies of data.] The appliction must return this 'loaned' data to the DataReader. The return_loan() routine indicates to the DataReader that the application no longer requires access to the data and sample_infos.

A call to **return_loan()** operation must be made only if previous **read()** or **take()** calls 'loaned' data to the application. See read() for a discussion of when data is 'loaned'.

If the **received_data** or **sample_infos** parameters provided do not identify data obtained from DataReader **dr,** then the error RETCODE_PRECONDITION_NOT_MET will be returned.

A DataReader cannot be deleted if any 'loans' are outstanding.

See also

com.toc.coredx.DDS.FooDataReader::read(FooSeq received_data, SampleInfoSeq sample_infos, int max_samples, long sample_states, long view_states, long instance_states) read()

3.25.2.10 ReturnCode_t take (FooSeq *received_data,* SampleInfoSeq *sample_infos,* int *max_samples,* long *sample_states,* long *view_states,* long *instance_states*)

This operation takes a collection of data values (with associated SampleInfo) from the DataReader. This routine, and the related **read()**, provide the interface for an application to access published data. There are several varieties of **read()** and **take()**, to facilitate different access patterns or approaches.

The primary difference between **read()** and **take()** is that **take()** removes all returned data samples from the DataReader while **read()** does not. Sequential **read()** calls will return the same data samples each time (if nothing else changes); while sequential **take()** calls will return data samples for only the first call. Subsequent **take()** calls will return an empty collection (if no new data arrives).

The specific behavior of **take()** depends on several things: input paramters, the QoS settings of the DataReader, and the state of recieved data. First, the **received_data** and **sample_infos** arguments affect the following:

- how many samples are returned, and
- whether the returned data should be 'loaned' or copied.

The argument **max_samples** is used to further limit the number of samples returned.

The **sample_states, view_states,** and **instance_states** arguments are used to selectively add data samples to the returned collections. These arguments indicate the desired 'states' for data samples and instances. These state arguments are bit masks; they can be the bit-wise OR of several individual state flags or they may use the special 'ANY' constants (e.g.: DDS.ANY_SAMPLE_STATE). Only samples that have a matching state for all three categories are added to the returned collection.

The order of samples in the returned collections is determined by the **PRESENTATION** and **DESTINATION_ORDER** QoS policies.

The returned collection is held in **recieved_data** and **samples_infos.** These two sequences operate together to represent a sequence of pairs (data, SampleInfo). Each data item in **received_data** has a corresponding entry in **sample_infos** that provides associated 'meta-data'. See SampleInfo for a description of this meta-data.

In CoreDX DDS, the returned sequences contain 'loaned' data. This provides zero-copy access to the data, and provides a very efficient data access mechanism. Becuase the data is 'loaned' to the application, the application is required to indicate when it is finished accessing the data. This is accomplished by calling return_loan().

The **take()** operation will set the SampleInfo.sample_state to DDS.READ_SAMPLE_STATE.

The **take()** operation may set the SampleInfo.view_state to DDS.NOT_NEW_VIEW_STATE, if a sample of the most recent generation of the instance is taken.

If there is no data found, then the **take()** will return RETCODE_NO_DATA.

This routine is data type specific. The generated type specific DataReader includes an implementation of this routine which should be used to support type-safety.

3.25.2.11 ReturnCode_t take_instance (FooSeq *received_data,* SampleInfoSeq *sample_infos,* int *max_samples,* InstanceHandle_t *handle,* long *sample_states,* long *view_states,* long *instance_states*)

This operation takes a collection of data values (with associated SampleInfo), belonging to a particular instance, from the DataReader.

This routine is data type specific. The generated type specific DataReader includes an implementation of this routine which should be used to support type-safety.

See also

com.toc.coredx.DDS.FooDataReader::take(FooSeq received_data, SampleInfoSeq sample_infos, int

max_samples, long sample_states, long view_states, long instance_states) take()

3.25.2.12 ReturnCode_t take_next_instance (FooSeq *received_data,* SampleInfoSeq *sample_infos,* int *max_samples,* InstanceHandle_t *prev_handle,* long *sample_states,* long *view_states,* long *instance_states*)

This operation takes a collection of data values (with associated SampleInfo), instance by instance, from the DataReader. To start,pass HANDLE_NIL for parameter prev_handle. After the last instance (in order) has been taken, this routine will return RETCODE_NO_DATA.

This routine is data type specific. The generated type specific DataReader includes an implementation of this routine which should be used to support type-safety.

See also

com.toc.coredx.DDS.FooDataReader::take(FooSeq received_data, SampleInfoSeq sample_infos, int max_samples, long sample_states, long view_states, long instance_states) take()

3.25.2.13 ReturnCode_t take_next_instance_w_condition (FooSeq *received_data,* SampleInfoSeq *sample_infos,* int *max_samples,* InstanceHandle_t *handle,* ReadCondition *condition*)

This operation takes a collection of data values (with associated SampleInfo), instance by instance subject to a filter, from the DataReader.

This routine is data type specific. The generated type specific DataReader includes an implementation of this routine which should be used to support type-safety.

See also

com.toc.coredx.DDS.FooDataReader::take(FooSeq received_data, SampleInfoSeq sample_infos, int max_samples, long sample_states, long view_states, long instance_states) take()

3.25.2.14 ReturnCode_t take_next_sample (Foo *received_data,* SampleInfo *sample_info*)

This operation takes a data value (with associated SampleInfo) from the DataReader.

This routine is data type specific. The generated type specific DataReader includes an implementation of this routine which should be used to support type-safety.

See also

com.toc.coredx.DDS.FooDataReader::take(FooSeq received_data, SampleInfoSeq sample_infos, int max_samples, long sample_states, long view_states, long instance_states) take()

3.25.2.15 ReturnCode_t take_w_condition (FooSeq *received_data,* SampleInfoSeq *sample_infos,* int *max_samples,* ReadCondition *condition*)

This operation takes a collection of data values (with associated SampleInfo), subject to a filter, from the DataReader.

This routine is data type specific. The generated type specific DataReader includes an implementation of this routine which should be used to support type-safety.

See also

com.toc.coredx.DDS.FooDataReader::take(FooSeq received_data, SampleInfoSeq sample_infos, int max_samples, long sample_states, long view_states, long instance_states) take()

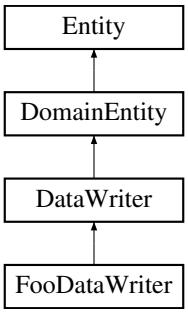
The documentation for this class was generated from the following file:

- /home/ctucker/coredx_v3.4rc/src/dds_java/com/toc/coredx/DDS/FooDataReader.java

3.26 FooDataWriter Class Reference

The FooDataWriter is an example specialized DataWriter (generated by the coredx_ddl compiler) for writing data samples of type 'Foo'.

Inheritance diagram for FooDataWriter:



Public Member Functions

- ReturnCode_t get_key_value (Foo key_holder, InstanceHandle_t handle)
- InstanceHandle_t lookup_instance (Foo instance_data)
- InstanceHandle_t register_instance (Foo instance_data)
- InstanceHandle_t register_instance_w_timestamp (Foo instance_data, Time_t ts)
- ReturnCode_t unregister_instance (Foo instance_data, InstanceHandle_t handle)
- ReturnCode_t unregister_instance_w_timestamp (Foo instance_data, InstanceHandle_t handle, Time_t timestamp)
- ReturnCode_t write (Foo instance_data, InstanceHandle_t handle)
- ReturnCode_t write_w_timestamp (Foo instance_data, InstanceHandle_t handle, Time_t timestamp)
- ReturnCode_t dispose (Foo instance_data, InstanceHandle_t handle)
- ReturnCode_t dispose_w_timestamp (Foo instance_data, InstanceHandle_t handle, Time_t timestamp)

3.26.1 Detailed Description

The FooDataWriter is an example specialized DataWriter (generated by the coredx_ddl compiler) for writing data samples of type 'Foo'. The FooDataWriter is an implementation of the base DataWriter class that has been extended to support the 'Foo' data type.

Note: the FooDataWriter is included here for documentation purposes only, and does not really exist in the com.toc.coredx.DDS package.

3.26.2 Member Function Documentation

3.26.2.1 ReturnCode_t dispose (Foo *instance_data,* InstanceHandle_t *handle*)

Indicates that the instance no longer exists. If **handle** is not HANDLE_NIL, then **handle** must identify a valid instance that has been previously registered or written by this DataWriter. The current time is used as the source timstamp.

3.26.2.2 ReturnCode_t dispose_w_timestamp (Foo *instance_data,* InstanceHandle_t *handle,* Time_t *timestamp*)

Indicates that the instance no longer exists. If **handle** is not HANDLE_NIL, then **handle** must identify a valid instance that has been previously registered or written by this DataWriter. The **source_timestamp** is used as the source timstamp for the published message.

3.26.2.3 ReturnCode_t get_key_value (Foo *key_holder,* InstanceHandle_t *handle*)

This routine will populate the data structure indicated by **key_holder** with the key infomation identified by **handle.**

This routine is data type specific. The generated type specific DataWriter includes an implementation of this routine which should be used to support type-safety.

3.26.2.4 InstanceHandle_t lookup_instance (Foo *instance_data*)

Returns the handle that identifies the data instance provided in **instance_data.** The 'key' field values of the data are associated with a unique handle.

3.26.2.5 InstanceHandle_t register_instance (Foo *instance_data*)

Declares the existance of an instance identified by the 'key fields' in **instance_data.** The handle that uniquely identifies the instance is returned. The current time is used as the source timestamp.

3.26.2.6 InstanceHandle_t register_instance_w_timestamp (Foo *instance_data,* Time_t *ts*)

Declares the existance of an instance identified by the 'key fields' in **instance_data.** The handle that uniquely identifies the instance is returned. The **source_timestamp** is used as the source timestamp.

3.26.2.7 ReturnCode_t unregister_instance (Foo *instance_data,* InstanceHandle_t *handle*)

Indicates that the writer will no longer be providing updates the specified instance. If **handle** is not HANDLE_NIL, then **handle** must identify a valid instance that has been previously registered or written

by this DataWriter. The current time is used as the source timstamp.

3.26.2.8 ReturnCode_t unregister_instance_w_timestamp (Foo *instance_data,* InstanceHandle_t *handle,* Time_t *timestamp*)

Indicates that the writer will no longer be providing updates the specified instance. If **handle** is not HANDLE_NIL, then **handle** must identify a valid instance that has been previously registered or written by this DataWriter. The provided **source_timestamp** is used as the source timstamp.

3.26.2.9 ReturnCode_t write (Foo *instance_data,* InstanceHandle_t *handle*)

Publishes the provided **instance_data.** If **handle** is HANDLE_NIL, then the handle is determined from the 'key fields' of **instance_data**.

The current time is used as the source timestamp for the published data.

This routine may block if RELIABILITY kind == RELIABLE, RESOURCE_LIMITS are not UNLIMITED, and the local resources are exhausted. The routine will block at most 'max_blocking_time' as configured in the RELIABLITY QoS. If the routine is still unable to handle the data, after blocking, then RETCODE_-TIMEOUT is returned.

The routine may return RETCODE_OUT_OF_RESOURCES if it is determined that no amount of blocking will allow the data to be processed.

On success, RETCODE_OK is returned.

3.26.2.10 ReturnCode_t write_w_timestamp (Foo *instance_data,* InstanceHandle_t *handle,* Time_t *timestamp*)

Publishes the provided **instance_data.** If **handle** is HANDLE_NIL, then the handle is determined from the 'key fields' of **instance_data.**

The **source_timestamp** is used as the source timestamp for the published data.

This routine may block if RELIABILITY kind == RELIABLE, RESOURCE_LIMITS are not UNLIMITED, and the local resources are exhausted. The routine will block at most 'max_blocking_time' as configured in the RELIABLITY QoS. If the routine is still unable to handle the data, after blocking, then RETCODE_-TIMEOUT is returned.

The routine may return RETCODE_OUT_OF_RESOURCES if it is determined that no amount of blocking will allow the data to be processed.

On success, RETCODE_OK is returned.

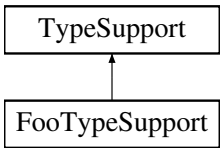
The documentation for this class was generated from the following file:

- /home/ctucker/coredx_v3.4rc/src/dds_java/com/toc/coredx/DDS/FooDataWriter.java

3.27 FooTypeSupport Class Reference

The FooTypeSupport class impliments the TypeSupport inteface for the Data Type 'Foo'.

Inheritance diagram for FooTypeSupport:



Public Member Functions

- ReturnCode_t register_type (DomainParticipant dp, String type_name)
- String get_type_name ()

3.27.1 Detailed Description

The FooTypeSupport class impliments the TypeSupport inteface for the Data Type 'Foo'. This provides a mechanism for an application to register the 'Foo' application defined data type with CoreDX DDS.

Note: FooTypeSupport is provided here for illustration purposes and is not really included in the com.toc.coredx.DDS package.

See also

com.toc.coredx.DDS.FooTypeSupport FooTypeSupport

3.27.2 Member Function Documentation

3.27.2.1 String get_type_name ( )

Returns the default name of the Data Type supported by this instance of TypeSupport (in this case, 'Foo').

Implements TypeSupport.

3.27.2.2 ReturnCode_t register_type (DomainParticipant *dp*, String *type_name*)

Registers the 'Foo' Data Type with the DDS infrastructure. The type is registered under the name 'type_-name'. If the 'type_name' parameter is null, then the default name ('Foo' in this case), will be used.

Implements TypeSupport.

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

- /home/ctucker/coredx_v3.4rc/src/dds_java/com/toc/coredx/DDS/FooTypeSupport.java

3.28 GroupDataQosPolicy Class Reference

Allows the application to attach arbitrary information to a Publisher or Subscriber.

3.28.1 Detailed Description

Allows the application to attach arbitrary information to a Publisher or Subscriber. The value of the GROUP_-DATA is propagated and made available to applications through the built-in topics.

The group data is implemented as a vector of Bytes, and can be used to store any application defined data.

This QoS can be used by an application in combination with the DataReaderListener and DataWriterListener to implement conditional matching policies.

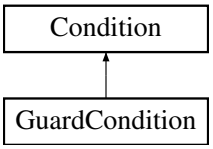
The documentation for this class was generated from the following file:

- /home/ctucker/coredx_v3.4rc/src/dds_java/com/toc/coredx/DDS/GroupDataQosPolicy.java

3.29 GuardCondition Class Reference

A GuardCondition is a Condition where the **trigger_value** is under application control.

Inheritance diagram for GuardCondition:



Public Member Functions

- GuardCondition ()

3.29.1 Detailed Description

A GuardCondition is a Condition where the **trigger_value** is under application control.

3.29.2 Constructor & Destructor Documentation

3.29.2.1 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_java/com/toc/coredx/DDS/GuardCondition.java

3.30 HistoryQosPolicy Class Reference

Controls the ammount of historical data maintained by a DataReader or DataWriter.

3.30.1 Detailed Description

Controls the ammount of historical data maintained by a DataReader or DataWriter. The history kind can be set to KEEP_ALL_HISTORY_QOS or KEEP_LAST_HISTORY_QOS.

If the history kind is KEEP_LAST, then the history depth field is used to determine the number of samples to keep. The default setting is KEEP_LAST with a depth of 1. This means that for a single sample (the most recent) is kept for each data instance.

If the history kind is KEEP_ALL, then the depth field is unused, and all samples will be kept, within the limits set by the ResourceLimits QoS policy.

This QoS policy must be consistent with the ResourceLimits policy.

See also

com.toc.coredx.DDS.ResourceLimitsQosPolicy

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

- /home/ctucker/coredx_v3.4rc/src/dds_java/com/toc/coredx/DDS/HistoryQosPolicy.java

3.31 InconsistentTopicStatus Class Reference

InconsistentTopicStatus provides status related to the on_inconsistent_topic listener methods of the TopicListener structure.

Public Member Functions

- int get_total_count ()
- int get_total_count_change ()

3.31.1 Detailed Description

InconsistentTopicStatus provides status related to the on_inconsistent_topic listener methods of the TopicListener structure.

3.31.2 Member Function Documentation

3.31.2.1 int get_total_count ( ) [inline]

Gets the **total_count** value. Cummulative count of the discovered Topics having a matching name and inconsistent characteristics.

3.31.2.2 int get_total_count_change ( ) [inline]

Gets the **total_count_change** value. Change in **total_count** since the last time the listener was called or status was read.

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

- /home/ctucker/coredx_v3.4rc/src/dds_java/com/toc/coredx/DDS/InconsistentTopicStatus.java

3.32 InstanceHandle_t Class Reference

Used to identify a DDS Entity or data instance. An InstanceHandle is unique within its context. For example, each data instance known by a DataReader is assigned a unique InstanceHandle. However, these handles may not be unique when compared to handles from another DataReader.

3.32.1 Detailed Description

Used to identify a DDS Entity or data instance. An InstanceHandle is unique within its context. For example, each data instance known by a DataReader is assigned a unique InstanceHandle. However, these handles may not be unique when compared to handles from another DataReader.

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

- /home/ctucker/coredx_v3.4rc/src/dds_java/com/toc/coredx/DDS/InstanceHandle_t.java

3.33 LatencyBudgetQosPolicy Class Reference

Specifies allowable latency.

3.33.1 Detailed Description

Specifies allowable latency. The latency budget is used internally by CoreDX DDS to optimize data messaging. If the middleware is provided a latency budget that is non-zero, then several internal operations can be optimized to reduce message and processing overhead.

To be compatible, the DataWriter must offer a latency_budget that is equal to or smaller than that requested by the DataReader.

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

- /home/ctucker/coredx_v3.4rc/src/dds_java/com/toc/coredx/DDS/LatencyBudgetQosPolicy.java

3.34 LifespanQosPolicy Class Reference

Specifies the maximum duration of validity of the data written by the DataWriter.

3.34.1 Detailed Description

Specifies the maximum duration of validity of the data written by the DataWriter. The default value of the lifespan duration is infinite.

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

- /home/ctucker/coredx_v3.4rc/src/dds_java/com/toc/coredx/DDS/LifespanQosPolicy.java

3.35 LivelinessChangedStatus Class Reference

Status related to the on_liveliness_changed listener methods of the DataReader, Subscriber, and DomainParticipant structures.

Public Member Functions

- int get_alive_count ()
- int get_alive_count_change ()
- int get_not_alive_count ()
- int get_not_alive_count_change ()
- InstanceHandle_t get_last_publication_handle ()

3.35.1 Detailed Description

Status related to the on_liveliness_changed listener methods of the DataReader, Subscriber, and DomainParticipant structures.

3.35.2 Member Function Documentation

3.35.2.1 int get_alive_count ( ) [inline]

Get the number of 'active' DataWriters matched to this DataReader.

3.35.2.2 int get_alive_count_change ( ) [inline]

Get the change in **alive_count** since the last time the listener was called or status was read.

3.35.2.3 InstanceHandle_t get_last_publication_handle ( ) [inline]

Get the handle identifying the most recent DataWriter whose liveliness changed.

3.35.2.4 int get_not_alive_count ( ) [inline]

Get the number of 'not-alive' DataWriters matched to this DataReader.

3.35.2.5 int get_not_alive_count_change ( ) [inline]

Get the change in **not_alive_count** since the last time the listener was called or status was read.

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

- /home/ctucker/coredx_v3.4rc/src/dds_java/com/toc/coredx/DDS/LivelinessChangedStatus.java

3.36 LivelinessLostStatus Class Reference

Status related to the on_liveliness_lost listener methods of the DataWriter, Publisher, and DomainParticipant structures.

Public Member Functions

- int get_total_count ()
- int get_total_count_change ()

3.36.1 Detailed Description

Status related to the on_liveliness_lost listener methods of the DataWriter, Publisher, and DomainParticipant structures.

3.36.2 Member Function Documentation

3.36.2.1 int get_total_count ( ) [inline]

Get the cummulative number of times that an 'alive' DataWriter became not alive.

3.36.2.2 int get_total_count_change ( ) [inline]

Get the change in **total_count** since the last time the listener was called or status was read.

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

- /home/ctucker/coredx_v3.4rc/src/dds_java/com/toc/coredx/DDS/LivelinessLostStatus.java

3.37 LivelinessQosPolicy Class Reference

Determines the mechanism and parameters used by the application to determine whether an Entity is alive.

3.37.1 Detailed Description

Determines the mechanism and parameters used by the application to determine whether an Entity is alive. The QoS defines the kind of liveliness (how liveliness is maintained) as well as a 'lease_duration' which specifies an interval for checking liveliness. The liveliness kind can be one of the following:

- AUTOMATIC
- MANUAL_BY_PARTICIPANT
- MANUAL_BY_TOPIC

The lease_duration indicates the maximum interval between liveliness indications from the publishing entity. If this period elapses with no indication from the publishing entity, then the entity is considered not alive.

Changes in liveliness are indicated to the application through status changes and the DataWriterListener.on_liveliness_lost and DataReaderListener.on_liveliness_changed listeners.

See also

com.toc.coredx.DDS.DataReaderListener::on_liveliness_changed(DataReader the_reader, LivelinessChangedStatus %DDS Status Structures) on_liveliness_changed
com.toc.coredx.DDS.DataWriterListener::on_liveliness_lost(DataWriter writer, LivelinessLostStatus %DDS Status Structures) on_liveliness_lost

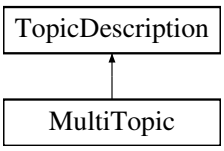
The documentation for this class was generated from the following file:

- /home/ctucker/coredx_v3.4rc/src/dds_java/com/toc/coredx/DDS/LivelinessQosPolicy.java

3.38 MultiTopic Class Reference

MultiTopic provides a topic that may include data from multiple Topics.

Inheritance diagram for MultiTopic:



Public Member Functions

- DomainParticipant get_participant ()
- String get_type_name ()
- String get_name ()

3.38.1 Detailed Description

MultiTopic provides a topic that may include data from multiple Topics.

Not Yet Supported

This is currently unsupported in the Java language binding.

3.38.2 Member Function Documentation

3.38.2.1 String get_name ( ) [inline]

Not Yet Supported

This routine is not yet implemented.

Implements TopicDescription.

3.38.2.2 DomainParticipant get_participant ( ) [inline]

Not Yet Supported

This routine is not yet implemented.

Implements TopicDescription.

3.38.2.3 String get_type_name ( ) [inline]

Not Yet Supported

This routine is not yet implemented.

Implements TopicDescription.

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

- /home/ctucker/coredx_v3.4rc/src/dds_java/com/toc/coredx/DDS/MultiTopic.java

3.39 OfferedDeadlineMissedStatus Class Reference

Status related to the on_offered_deadline_missed listener methods of the DataWriter, Publisher, and DomainParticipant structures.

Public Member Functions

- int get_total_count ()
- int get_total_count_change ()
- InstanceHandle_t get_last_instance_handle ()

3.39.1 Detailed Description

Status related to the on_offered_deadline_missed listener methods of the DataWriter, Publisher, and DomainParticipant structures.

3.39.2 Member Function Documentation

3.39.2.1 InstanceHandle_t get_last_instance_handle ( ) [inline]

Get the handle identifying the most recent instance whose deadline was missed.

3.39.2.2 int get_total_count ( ) [inline]

Get the cummulative count of the number of deadlines missed by this DataWriter.

3.39.2.3 int get_total_count_change ( ) [inline]

Get the change in **total_count** since the last time the listener was called or status was read.

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

- /home/ctucker/coredx_v3.4rc/src/dds_java/com/toc/coredx/DDS/OfferedDeadlineMissedStatus.java

3.40 OfferedIncompatibleQosStatus Class Reference

Status related to the on_offered_incompatible_qos listener methods of the DataWriter, Publisher, and DomainParticipant structures.

Public Member Functions

- int get_total_count ()
- int get_total_count_change ()
- int get_last_policy_id ()
- Vector get_policies ()

3.40.1 Detailed Description

Status related to the on_offered_incompatible_qos listener methods of the DataWriter, Publisher, and DomainParticipant structures.

3.40.2 Member Function Documentation

3.40.2.1 int get_last_policy_id ( ) [inline]

Get the **id** of the most recent requested incompatible QoS policy. Convert **id** to a destcriptive string with DDS.qos_policy_str(int qos_policy_id).

3.40.2.2 Vector get_policies ( ) [inline]

A list of QoS policies and the total number of times each QoS policy was found to be incompatible. Vector elements are of type QosPolicyCount.

3.40.2.3 int get_total_count ( ) [inline]

Get the cummulative count of the number of DataWriters discovered having matching Topic and incompatible QoS.

3.40.2.4 int get_total_count_change ( ) [inline]

Get the change in **total_count** since the last time the listener was called or status was read.

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

- /home/ctucker/coredx_v3.4rc/src/dds_java/com/toc/coredx/DDS/OfferedIncompatibleQosStatus.java

3.41 OwnershipQosPolicy Class Reference

Determines instance ownership in the case of multple writers. CoreDX DDS supports both SHARED_-OWNERSHIP_QOS and EXCLUSIVE_OWNERSHIP_QOS.

3.41.1 Detailed Description

Determines instance ownership in the case of multple writers. CoreDX DDS supports both SHARED_-OWNERSHIP_QOS and EXCLUSIVE_OWNERSHIP_QOS.

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

- /home/ctucker/coredx_v3.4rc/src/dds_java/com/toc/coredx/DDS/OwnershipQosPolicy.java

3.42 OwnershipStrengthQosPolicy Class Reference

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.

3.42.1 Detailed Description

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.

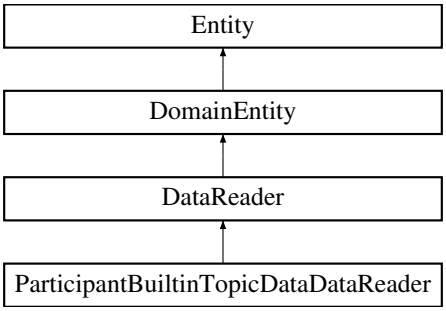
The documentation for this class was generated from the following file:

- /home/ctucker/coredx_v3.4rc/src/dds_java/com/toc/coredx/DDS/OwnershipStrengthQosPolicy.java

3.43 ParticipantBuiltinTopicDataDataReader Class Reference

The ParticipantBuiltinTopicDataDataReader is a built-in DataReader that can be used to access Participant Discovery information. The reader can be accessed through the special 'builtin' Subscriber via the DomainParticipant.get_builtin_subscriber() routine.

Inheritance diagram for ParticipantBuiltinTopicDataDataReader:



3.43.1 Detailed Description

The ParticipantBuiltinTopicDataDataReader is a built-in DataReader that can be used to access Participant Discovery information. The reader can be accessed through the special 'builtin' Subscriber via the DomainParticipant.get_builtin_subscriber() routine.

See also

FooDataReader
ParticipantBuiltinTopicData
DomainParticipant::get_builtin_subscriber

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

- /home/ctucker/coredx_v3.4rc/src/dds_java/com/toc/coredx/DDS/ParticipantBuiltinTopicDataDataReader.java

3.44 PartitionQosPolicy Class Reference

3.44.1 Detailed Description

Defines a logical data partition. This QoS is assigned to a Publisher or Subscriber. DataWriters and DataReaders exist within the partition defined by their parent. DataWriters and DataReaders communicate only if they have a matching partition (and all other QoS and topic parameters match).

The partition QoS can be changed dynamically. Dynamic changes to partition may cause new matches between DataWriters and DataReaders or may break existing matches.

The partition QoS comprises a vector of Strings. Two Partitions 'match' if any string in one matches any string in the other.

A partition string may be a regular expression with wildcards as defined by POSIX fnmatch API (1003.2-1992 section B.6).

An empty Partition, that is a Partition with an empty name vector, is treated as a Partition with a single empty string of "".

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

- /home/ctucker/coredx_v3.4rc/src/dds_java/com/toc/coredx/DDS/PartitionQosPolicy.java

3.45 PresentationQosPolicy Class Reference

Controls the presentation of received data samples to the application. CoreDX DDS currently supports only the access_scope = INSTANCE_PRESENTATION_QOS policy.

Public Attributes

- PresentationQosPolicyAccessScopeKind access_scope

  the 'scope' of the presentation policy. Determines the extent to which the sample 'coherency' and 'order' are maintained.

- boolean coherent_access

  Determines if coherent access is supported within the defined 'scope'.

- boolean ordered_access

  Determines if ordered access is supported within the defined 'scope'.

3.45.1 Detailed Description

Controls the presentation of received data samples to the application. CoreDX DDS currently supports only the access_scope = INSTANCE_PRESENTATION_QOS policy.

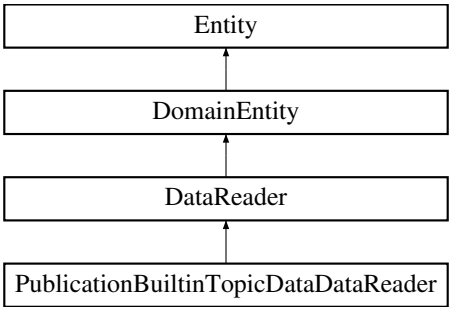
The documentation for this class was generated from the following file:

- /home/ctucker/coredx_v3.4rc/src/dds_java/com/toc/coredx/DDS/PresentationQosPolicy.java

3.46 PublicationBuiltinTopicDataDataReader Class Reference

The PublicationBuiltinTopicDataDataReader is a built-in DataReader that can be used to access Publication Discovery information. The reader can be accessed through the special 'builtin' Subscriber via the DomainParticipant.get_builtin_subscriber() routine.

Inheritance diagram for PublicationBuiltinTopicDataDataReader:



3.46.1 Detailed Description

The PublicationBuiltinTopicDataDataReader is a built-in DataReader that can be used to access Publication Discovery information. The reader can be accessed through the special 'builtin' Subscriber via the DomainParticipant.get_builtin_subscriber() routine.

See also

FooDataReader
PublicationBuiltinTopicData
DomainParticipant::get_builtin_subscriber

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

- /home/ctucker/coredx_v3.4rc/src/dds_java/com/toc/coredx/DDS/PublicationBuiltinTopicDataDataReader.java

3.47 PublicationMatchedStatus Class Reference

Status related to the on_publication_matched listener methods of the DataWriter, Publisher, and DomainParticipant structures.

Public Member Functions

- int get_total_count ()
- int get_total_count_change ()
- int get_current_count ()
- int get_current_count_change ()
- InstanceHandle_t get_last_subscription_handle ()

3.47.1 Detailed Description

Status related to the on_publication_matched listener methods of the DataWriter, Publisher, and DomainParticipant structures.

3.47.2 Member Function Documentation

3.47.2.1 int get_current_count ( ) [inline]

Get the current number of DataReaders matched to the DataWriter.

3.47.2.2 int get_current_count_change ( ) [inline]

Get the change in **current_count** since the last time the listener was called or status was read.

3.47.2.3 InstanceHandle_t get_last_subscription_handle ( ) [inline]

Get the handle of the most recently matched DataReader (subscription).

3.47.2.4 int get_total_count ( ) [inline]

Get the cummulative count of the number of times this DataWriter has discovered a matching DataReader.

3.47.2.5 int get_total_count_change ( ) [inline]

Get the change in **total_count** since the last time the listener was called or status was read.

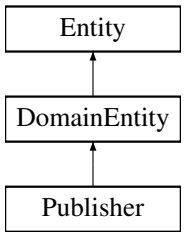
The documentation for this class was generated from the following file:

- /home/ctucker/coredx_v3.4rc/src/dds_java/com/toc/coredx/DDS/PublicationMatchedStatus.java

3.48 Publisher Class Reference

The Publisher configures, creates, manages and destroys DataWriters.

Inheritance diagram for Publisher:



Public Member Functions

- ReturnCode_t enable ()
- DomainParticipant get_participant ()
- DataWriter create_datawriter (Topic topic, DataWriterQos qos, DataWriterListener listener, long 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, long 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)

3.48.1 Detailed Description

The Publisher configures, creates, manages and destroys DataWriters.

3.48.2 Member Function Documentation

3.48.2.1 ReturnCode_t begin_coherent_changes ( ) [inline]

Not Yet Supported

This operation is not yet implemented.

3.48.2.2 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.

3.48.2.3 DataWriter create_datawriter (Topic *topic*, DataWriterQos *qos*, DataWriterListener *listener*, long *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.**

3.48.2.4 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.

3.48.2.5 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.

3.48.2.6 ReturnCode_t enable ( ) [inline]

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.

3.48.2.7 ReturnCode_t end_coherent_changes ( ) [inline]

Not Yet Supported

This operation is not yet implemented.

3.48.2.8 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.

3.48.2.9 PublisherListener get_listener ( ) [inline]

This operation returns the currently installed PublisherListener.

3.48.2.10 DomainParticipant get_participant ( ) [inline]

This operation returns the DomainParticipant this Publisher belongs to.

3.48.2.11 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.

3.48.2.12 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.**

3.48.2.13 ReturnCode_t resume_publications ( ) [inline]

Not Yet Supported

This operation is not yet implemented.

3.48.2.14 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.

3.48.2.15 ReturnCode_t set_listener (PublisherListener *new_listener,* long *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.

3.48.2.16 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.

3.48.2.17 ReturnCode_t suspend_publications ( ) [inline]

Not Yet Supported

This operation is not yet implemented.

3.48.2.18 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_java/com/toc/coredx/DDS/Publisher.java

3.49 PublisherListener Interface Reference

Public Member Functions

- void on_offered_incompatible_qos (DataWriter writer, OfferedIncompatibleQosStatus status)
- void on_liveliness_lost (DataWriter writer, LivelinessLostStatus status)
- void on_publication_matched (DataWriter writer, PublicationMatchedStatus status)
- long get_nil_mask ()

Package Functions

- void on_offered_deadline_missed (DataWriter writer, OfferedDeadlineMissedStatus status)

3.49.1 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().

3.49.2 Member Function Documentation

3.49.2.1 long get_nil_mask ( )

get_nil_mask() returns a bitmask indicating which listener methods (if any) should be considered NIL, and therefore, should not be invoked.

3.49.2.2 void on_liveliness_lost (DataWriter *writer,* LivelinessLostStatus *status*)

Called when the CoreDX infrastructure detects that a DataWriter contained in the Publisher has not satisfied its LIVELINESS QoS setting. This listener is invoked only if the concerned DataWriter does not have an **on_offered_deadline_missed** listener installed.

The **status** argument provides a snapshot of the status at the time the listener was invoked.

3.49.2.3 void on_offered_deadline_missed (DataWriter *writer,* OfferedDeadlineMissedStatus *status*) [package]

Called when the CoreDX infrastructure detects that a DataWriter contained in the Publisher has failed to meet its DEADLINE QoS commitment. This listener is invoked only if the concerned DataWriter does not have an **on_offered_deadline_missed** listener installed.

The **status** argument provides a snapshot of the status at the time the listener was invoked.

3.49.2.4 void on_offered_incompatible_qos (DataWriter *writer,* OfferedIncompatibleQosStatus *status*)

Called when the CoreDX infrastructure detects that a DataWriter contained in the Publisher has offered a QoS policy setting that is incompatible with that requested by a potentially matching DataReader. This listener is invoked only if the concerned DataWriter does not have an **on_offered_deadline_missed** listener installed.

The **status** argument provides a snapshot of the status at the time the listener was invoked.

3.49.2.5 void on_publication_matched (DataWriter *writer,* PublicationMatchedStatus *status*)

Called when the CoreDX infrastructure detects that a DataWriter contained in the Publisher has matched with a DataReader or has ceased to be matched with a DataReader. This listener is invoked only if the concerned DataWriter does not have an **on_offered_deadline_missed** listener installed.

The **status** argument provides a snapshot of the status at the time the listener was invoked.

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

- /home/ctucker/coredx_v3.4rc/src/dds_java/com/toc/coredx/DDS/PublisherListener.java

3.50 PublisherQos Class Reference

Structure that holds Publisher Quality of Service policies.

Public Attributes

- PresentationQosPolicy presentation
- PartitionQosPolicy partition
- GroupDataQosPolicy group_data
- EntityFactoryQosPolicy entity_factory

3.50.1 Detailed Description

Structure that holds Publisher Quality of Service policies.

See also

Publisher::set_qos(PublisherQos) set_qos()
Publisher::get_qos(PublisherQos) get_qos()
DomainParticipant::create_publisher(PublisherQos qos, PublisherListener listener, long mask) create_publisher()
DomainParticipant::set_default_publisher_qos(PublisherQos) set_default_publisher_qos()
DomainParticipant::get_default_publisher_qos(PublisherQos) get_default_publisher_qos()

3.50.2 Member Data Documentation

3.50.2.1 EntityFactoryQosPolicy entity_factory

Controls the behavior of the Publisher.create_datawriter() operation.

3.50.2.2 GroupDataQosPolicy group_data

A sequence of octets associated with the Publisher.

3.50.2.3 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.

3.50.2.4 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()

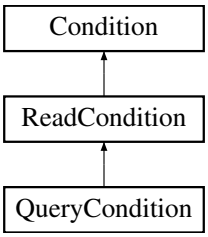
The documentation for this class was generated from the following file:

- /home/ctucker/coredx_v3.4rc/src/dds_java/com/toc/coredx/DDS/PublisherQos.java

3.51 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.

- Vector< String > get_query_parameters ()

  Provides access to the parameters of the query expression.

- ReturnCode_t set_query_parameters (Vector parameters)

  Modifies the parameters of the query expression.

3.51.1 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.

3.51.2 Member Function Documentation

3.51.2.1 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()

3.51.2.2 Vector<String> get_query_parameters ( ) [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()

3.51.2.3 ReturnCode_t set_query_parameters (Vector *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 refering to paramters in a query_-expression 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.

This routine allows the parameters to be changed dynamically.

See also

DataReader::create_querycondition()

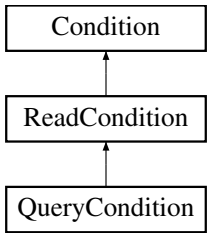
The documentation for this class was generated from the following file:

- /home/ctucker/coredx_v3.4rc/src/dds_java/com/toc/coredx/DDS/QueryCondition.java

3.52 ReadCondition Class Reference

A ReadCondition is a specialized Condition associated with a DataReader.

Inheritance diagram for ReadCondition:



Public Member Functions

- long get_sample_state_mask ()
- long get_view_state_mask ()
- long get_instance_state_mask ()
- DataReader get_datareader ()

3.52.1 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)

3.52.2 Member Function Documentation

3.52.2.1 DataReader get_datareader ( ) [inline]

Gets the single DataReader associated with this ReadCondition.

3.52.2.2 long get_instance_state_mask ( ) [inline]

Gets the current value of the instance_state_mask in this ReadCondition.

3.52.2.3 long get_sample_state_mask ( ) [inline]

Gets the current value of the sample_state_mask in this ReadCondition.

3.52.2.4 long 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_java/com/toc/coredx/DDS/ReadCondition.java

3.53 ReaderDataLifecycleQosPolicy Class Reference

Specifies the lifecycle behavior of data instances managed by the DataReader.

3.53.1 Detailed Description

Specifies the lifecycle behavior of data instances managed by the DataReader. **autopurge_nowriter_samples_delay** indicates how long the DataReader will maintain instance samples that have instance_state NOT_ALIVE_NO_WRITERS.

autopurge_disposed_samples_delay indicates how long the DataReader will retain information about instances that have instance_state NOT_ALIVE_DISPOSED.

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

- /home/ctucker/coredx_v3.4rc/src/dds_java/com/toc/coredx/DDS/ReaderDataLifecycleQosPolicy.java

3.54 ReliabilityQosPolicy Class Reference

Indicates the level of reliability offered/provided by the Entity. If kind is RELIABLE_RELIABILITY_QOS, then the middleware will attempt to deliver all samples in the history cache. If samples are not received, then they will be retried.

3.54.1 Detailed Description

Indicates the level of reliability offered/provided by the Entity. If kind is RELIABLE_RELIABILITY_QOS, then the middleware will attempt to deliver all samples in the history cache. If samples are not received, then they will be retried. A kind of BEST_EFFORT_RELIABILITY_QOS indicates that the middleware does not need to retry delivery of data samples.

The samples available in the history cache are affected by the HistoryQosPolicy, ResourceLimitsQosPolicy, and the DurabilityQosPolicy.

Note: If samples are removed from the history cache, then they are not be available to be resent.

See also

com.toc.coredx.DDS.HistoryQosPolicy
com.toc.coredx.DDS.ResourceLimitsQosPolicy
com.toc.coredx.DDS.DurabilityQosPolicy

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

- /home/ctucker/coredx_v3.4rc/src/dds_java/com/toc/coredx/DDS/ReliabilityQosPolicy.java

3.55 RequestedDeadlineMissedStatus Class Reference

Status related to the on_requested_deadline_missed listener methods of the DataReader, Subscriber, and DomainParticipant structures.

Public Member Functions

- int get_total_count ()
- int get_total_count_change ()
- InstanceHandle_t get_last_instance_handle ()

3.55.1 Detailed Description

Status related to the on_requested_deadline_missed listener methods of the DataReader, Subscriber, and DomainParticipant structures.

3.55.2 Member Function Documentation

3.55.2.1 InstanceHandle_t get_last_instance_handle ( ) [inline]

Get the handle identifying the most recent instance whose deadline was missed.

3.55.2.2 int get_total_count ( ) [inline]

Get the cummulative count of the number of detected deadline misses for any instance read by the DataReader.

3.55.2.3 int get_total_count_change ( ) [inline]

Get the change in **total_count** since the last time the listener was called or status was read.

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

- /home/ctucker/coredx_v3.4rc/src/dds_java/com/toc/coredx/DDS/RequestedDeadlineMissedStatus.java

3.56 RequestedIncompatibleQosStatus Class Reference

Public Member Functions

- int get_total_count ()
- int get_total_count_change ()
- int get_last_policy_id ()
- Vector get_policies ()

3.56.1 Detailed Description

Status related to the on_requested_incompatible_qos listener methods of the DDS_DataReader, DDS_-Subscriber, and DDS_DomainParticipant structures.

3.56.2 Member Function Documentation

3.56.2.1 int get_last_policy_id ( ) [inline]

Get the **id** identifying the most recent QoS policy detected to be incompatible. Convert **id** to a destcriptive string with DDS.qos_policy_str(int qos_policy_id).

3.56.2.2 Vector get_policies ( ) [inline]

Get a list of QoS policies and the total number of times each QoS policy was found to be incompatible. Vector elements are of type QosPolicyCount.

3.56.2.3 int get_total_count ( ) [inline]

Get the cummulative count of the number of discovered DataReaders having matching Topic and incompatible QoS.

3.56.2.4 int get_total_count_change ( ) [inline]

Get the change in **total_count** since the last time the listener was called or status was read.

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

- /home/ctucker/coredx_v3.4rc/src/dds_java/com/toc/coredx/DDS/RequestedIncompatibleQosStatus.java

3.57 ResourceLimitsQosPolicy Class Reference

Specifies the resources that the Service can use to maintain data samples and instances.

3.57.1 Detailed Description

Specifies the resources that the Service can use to maintain data samples and instances.

See also

com.toc.coredx.DDS.ReliabilityQosPolicy
com.toc.coredx.DDS.HistoryQosPolicy

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

- /home/ctucker/coredx_v3.4rc/src/dds_java/com/toc/coredx/DDS/ResourceLimitsQosPolicy.java

3.58 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

- long sample_state
- long view_state
- long instance_state
- Time_t source_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
- boolean valid_data

3.58.1 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.

3.58.2 Member Data Documentation

3.58.2.1 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.

3.58.2.2 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.)

3.58.2.3 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.

3.58.2.4 InstanceHandle_t instance_handle

The handle that locally identifies the associated instance.

3.58.2.5 long 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) ...

3.58.2.6 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.)

3.58.2.7 InstanceHandle_t publication_handle

The local handle of the source DataWriter.

3.58.2.8 int sample_rank

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

3.58.2.9 long 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.

3.58.2.10 Time_t source_timestamp

The time provided by the DataWriter when the sample was written.

3.58.2.11 boolean 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).

3.58.2.12 long 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_java/com/toc/coredx/DDS/SampleInfo.java

3.59 SampleLostStatus Class Reference

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

Public Member Functions

- int get_total_count ()
- int get_total_count_change ()

3.59.1 Detailed Description

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

3.59.2 Member Function Documentation

3.59.2.1 int get_total_count ( ) [inline]

Gets the **total_count**. Cummulative count of all samples lost under the Topic.

3.59.2.2 int get_total_count_change ( ) [inline]

Gets the **total_count_change**. Change in **total_count** since the last time the listener was called or status was read.

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

- /home/ctucker/coredx_v3.4rc/src/dds_java/com/toc/coredx/DDS/SampleLostStatus.java

3.60 SampleRejectedStatus Class Reference

Status related to the on_sample_rejected listener methods of the DataReader, Subscriber, and DomainParticipant structures.

Public Member Functions

- int get_total_count ()
- int get_total_count_change ()
- SampleRejectedStatusKind get_last_reason ()
- InstanceHandle_t get_last_instance_handle ()

3.60.1 Detailed Description

Status related to the on_sample_rejected listener methods of the DataReader, Subscriber, and DomainParticipant structures.

3.60.2 Member Function Documentation

3.60.2.1 InstanceHandle_t get_last_instance_handle ( ) [inline]

Get the handle of the instance associated with the most recently rejected sample.

3.60.2.2 SampleRejectedStatusKind get_last_reason ( ) [inline]

Get the reason for rejecting the most recently rejected sample.

3.60.2.3 int get_total_count ( ) [inline]

Get the cummulative count of all samples rejected under the Topic.

3.60.2.4 int get_total_count_change ( ) [inline]

Get the change in **total_count** since the last time the listener was called or status was read.

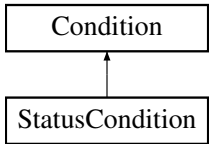
The documentation for this class was generated from the following file:

- /home/ctucker/coredx_v3.4rc/src/dds_java/com/toc/coredx/DDS/SampleRejectedStatus.java

3.61 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 (long mask)
- long get_enabled_statuses ()
- Entity get_entity ()

3.61.1 Detailed Description

A StatusCondition is a condition associated with an Entity. The **trigger_value** is driven by the communication status of the associated Entity.

3.61.2 Member Function Documentation

3.61.2.1 long 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

3.61.2.2 Entity get_entity ( ) [inline]

This routine returns the single entity associated with this StatusCondition.

3.61.2.3 ReturnCode_t set_enabled_statuses (long *mask*) [inline]

This routine sets the statuses which are enabled in this StatusCondition. The statuses are provided as a bitmask.

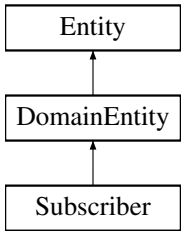
The documentation for this class was generated from the following file:

- /home/ctucker/coredx_v3.4rc/src/dds_java/com/toc/coredx/DDS/StatusCondition.java

3.62 Subscriber Class Reference

The Subscriber configures, creates, manages and destroys DataReaders.

Inheritance diagram for Subscriber:



Public Member Functions

- ReturnCode_t enable ()
- DataReader create_datareader (TopicDescription topic, DataReaderQos qos, DataReaderListener listener, long mask)
- ReturnCode_t delete_datareader (DataReader datareader)
- ReturnCode_t delete_contained_entities ()
- DataReader lookup_datareader (String topic_name)
- ReturnCode_t get_datareaders (Vector 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, long 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)

3.62.1 Detailed Description

The Subscriber configures, creates, manages and destroys DataReaders.

3.62.2 Member Function Documentation

3.62.2.1 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 Java language binding.

3.62.2.2 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.

3.62.2.3 DataReader create_datareader (TopicDescription *topic,* DataReaderQos *qos,* DataReaderListener *listener,* long *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.**

3.62.2.4 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.

3.62.2.5 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_-.tRETCODE_PRECONDITION_NOT_MET.

3.62.2.6 ReturnCode_t enable ( ) [inline]

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.

3.62.2.7 ReturnCode_t end_access ( ) [inline]

This operation closes a corresponding Subscriber.begin_access().

Not Yet Supported

This is currently unsupported in the Java language binding.

3.62.2.8 ReturnCode_t get_datareaders (Vector *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 Java language binding.

3.62.2.9 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.

3.62.2.10 SubscriberListener get_listener ( ) [inline]

This operation returns the currently installed SubscriberListener.

3.62.2.11 DomainParticipant get_participant ( ) [inline]

This operation returns the DomainParticipant this Subscriber belongs to.

3.62.2.12 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.

3.62.2.13 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.

3.62.2.14 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.

3.62.2.15 ReturnCode_t set_listener (SubscriberListener *new_listener,* long *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.

3.62.2.16 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_java/com/toc/coredx/DDS/Subscriber.java

3.63 SubscriberListener Interface Reference

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().

Public Member Functions

- void on_requested_incompatible_qos (DataReader the_reader, RequestedIncompatibleQosStatus status)
- void on_sample_rejected (DataReader the_reader, SampleRejectedStatus status)
- void on_liveliness_changed (DataReader the_reader, LivelinessChangedStatus status)
- void on_data_available (DataReader the_reader)
- void on_subscription_matched (DataReader the_reader, SubscriptionMatchedStatus status)
- void on_sample_lost (DataReader the_reader, SampleLostStatus status)
- void on_data_on_readers (Subscriber the_subscriber)
- long get_nil_mask ()

Package Functions

- void on_requested_deadline_missed (DataReader the_reader, RequestedDeadlineMissedStatus status)

3.63.1 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().

3.63.2 Member Function Documentation

3.63.2.1 long get_nil_mask ( )

get_nil_mask() returns a bitmask indicating which listener methods (if any) should be considered NIL, and therefore, should not be invoked.

3.63.2.2 void on_data_available (DataReader *the_reader*)

Called when the CoreDX infrastructure detects that a DataReader contained in the Subscriber has new data or data state information available. This listener is invoked only if the concerned DataReader does not have an **on_data_available** listener installed.

The **status** argument provides a snapshot of the status at the time the listener was invoked.

3.63.2.3 void on_data_on_readers (Subscriber *the_subscriber*)

Called when the CoreDX infrastructure detects that data or data state information arrives at any DataReader contained within the Subscriber.

The **status** argument provides a snapshot of the status at the time the listener was invoked.

3.63.2.4 void on_liveliness_changed (DataReader *the_reader,* LivelinessChangedStatus *status*)

Called when the CoreDX infrastructure detects that the liveliness of a DataWriter, matched to a DataReader within this Subscriber, has changed (either 'active' or 'inactive'). This listener is invoked only if the concerned DataReader does not have an **on_liveliness_changed** listener installed.

The **status** argument provides a snapshot of the status at the time the listener was invoked.

3.63.2.5 void on_requested_deadline_missed (DataReader *the_reader,* RequestedDeadlineMissedStatus *status*) `[package]`

Called when the CoreDX infrastructure detects that the QoS DEADLINE policy of a DataReader, contained in this Subscriber, was not satisfied for a data instance. This listener is invoked only if the concerned DataReader does not have an **on_requested_deadline_missed** listener installed.

The **status** argument provides a snapshot of the status at the time the listener was invoked.

3.63.2.6 void on_requested_incompatible_qos (DataReader *the_reader,* RequestedIncompatibleQosStatus *status*)

on_requested_incompatible_qos() is called when the CoreDX infrastructure detects that a DataReader contained in the Subscriber has requested a QoS policy that was incompatible with that offered by a DataWriter. This listener is invoked only if the concerned DataReader does not have an **on_requested_incompatible_qos** listener installed.

The **status** argument provides a snapshot of the status at the time the listener was invoked.

3.63.2.7 void on_sample_lost (DataReader *the_reader,* SampleLostStatus *status*)

Called when the CoreDX infrastructure detects that a DataReader contained in the Subscriber has lost a sample (never received). This listener is invoked only if the concerned DataReader does not have an **on_sample_lost** listener installed.

The **status** argument provides a snapshot of the status at the time the listener was invoked.

3.63.2.8 void on_sample_rejected (DataReader *the_reader*, SampleRejectedStatus *status*)

Called when the CoreDX infrastructure detects that a DataReader contained in the Subscriber has rejected a sample. This listener is invoked only if the concerned DataReader does not have an **on_sample_rejected** listener installed.

The **status** argument provides a snapshot of the status at the time the listener was invoked.

3.63.2.9 void on_subscription_matched (DataReader *the_reader*, SubscriptionMatchedStatus *status*)

Called when the CoreDX infrastructure detects that a DataReader contained in the Subscriber has matched or ceased to be matched with a DataWriter. This listener is invoked only if the concerned DataReader does not have an **on_subscription_matched** listener installed.

The **status** argument provides a snapshot of the status at the time the listener was invoked.

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

- /home/ctucker/coredx_v3.4rc/src/dds_java/com/toc/coredx/DDS/SubscriberListener.java

3.64 SubscriberQos Class Reference

Structure that holds DDS_Subscriber Quality of Service policies.

Public Attributes

- PresentationQosPolicy presentation
- PartitionQosPolicy partition
- GroupDataQosPolicy group_data
- EntityFactoryQosPolicy entity_factory

3.64.1 Detailed Description

Structure that holds DDS_Subscriber Quality of Service policies.

See also

Subscriber::set_qos(SubscriberQos)
Subscriber::get_qos(SubscriberQos)
DomainParticipant::create_subscriber(SubscriberQos qos, SubscriberListener listener, long mask)
DomainParticipant::set_default_subscriber_qos(SubscriberQos)
DomainParticipant::get_default_subscriber_qos(SubscriberQos)

3.64.2 Member Data Documentation

3.64.2.1 EntityFactoryQosPolicy entity_factory

Controls the behavior of the Subscriber.create_datareader() operation.

3.64.2.2 GroupDataQosPolicy group_data

A sequence of octets associated with the Publisher.

3.64.2.3 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.

3.64.2.4 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()

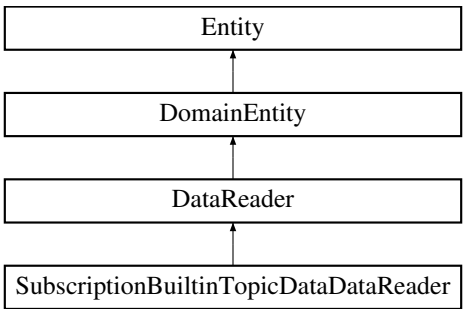
The documentation for this class was generated from the following file:

- /home/ctucker/coredx_v3.4rc/src/dds_java/com/toc/coredx/DDS/SubscriberQos.java

3.65 SubscriptionBuiltinTopicDataDataReader Class Reference

The SubscriptionBuiltinTopicDataDataReader is a built-in DataReader that can be used to access Subscription Discovery information.

Inheritance diagram for SubscriptionBuiltinTopicDataDataReader:



3.65.1 Detailed Description

The SubscriptionBuiltinTopicDataDataReader is a built-in DataReader that can be used to access Subscription Discovery information. The reader can be accessed through the special 'builtin' Subscriber via the DomainParticipant.get_builtin_subscriber() routine.

See also

FooDataReader
SubscriptionBuiltinTopicData
DomainParticipant::get_builtin_subscriber

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

- /home/ctucker/coredx_v3.4rc/src/dds_java/com/toc/coredx/DDS/SubscriptionBuiltinTopicDataDataReader.java

3.66 SubscriptionMatchedStatus Class Reference

Status related to the on_subscription_matched listener methods of the DataReader, Subscriber, and DomainParticipant structures.

Public Member Functions

- int get_total_count ()
- int get_total_count_change ()
- int get_current_count ()
- int get_current_count_change ()
- InstanceHandle_t get_last_publication_handle ()

3.66.1 Detailed Description

Status related to the on_subscription_matched listener methods of the DataReader, Subscriber, and DomainParticipant structures.

3.66.2 Member Function Documentation

3.66.2.1 int get_current_count ( ) [inline]

Get the current number of DataWriters matched to the DataReader.

3.66.2.2 int get_current_count_change ( ) [inline]

Get the change in **current_count** since the last time the listener was called or status was read.

3.66.2.3 InstanceHandle_t get_last_publication_handle ( ) [inline]

Get the handle of most recent matched DataWriter (publication).

3.66.2.4 int get_total_count ( ) [inline]

Get the cummulative count of the number of times this DataReader has discovered a matching DataWriter.

3.66.2.5 int get_total_count_change ( ) [inline]

Get the change in **total_count** since the last time the listener was called or status was read.

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

- /home/ctucker/coredx_v3.4rc/src/dds_java/com/toc/coredx/DDS/SubscriptionMatchedStatus.java

3.67 Time_t Class Reference

Represents time with nanosecond resolution.

3.67.1 Detailed Description

Represents time with nanosecond resolution.

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

- /home/ctucker/coredx_v3.4rc/src/dds_java/com/toc/coredx/DDS/Time_t.java

3.68 TimeBasedFilterQosPolicy Class Reference

Defines a filter based on time between samples. The DataReader indicates that it wants at most one sample for each instance every minimum_separation interval.

3.68.1 Detailed Description

Defines a filter based on time between samples. The DataReader indicates that it wants at most one sample for each instance every minimum_separation interval. It is inconsistent for a DataReader to have a minimum_separation larger than its DEADLINE period. By default minimum_separation=0 indicating that the DataReader should be sent all values.

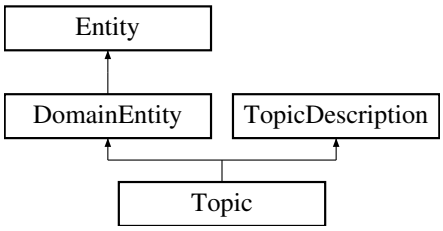
The documentation for this class was generated from the following file:

- /home/ctucker/coredx_v3.4rc/src/dds_java/com/toc/coredx/DDS/TimeBasedFilterQosPolicy.java

3.69 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

- ReturnCode_t enable ()
- 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.*

- ReturnCode_t set_qos (TopicQos qos)
- ReturnCode_t get_qos (TopicQos qos)
- ReturnCode_t set_listener (TopicListener new_listener, long mask)
- TopicListener get_listener ()
- ReturnCode_t get_inconsistent_topic_status (InconsistentTopicStatus status)

3.69.1 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.].

3.69.2 Member Function Documentation

3.69.2.1 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].

3.69.2.2 ReturnCode_t get_inconsistent_topic_status (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.

3.69.2.3 TopicListener get_listener ( ) [inline]

This operation returns the currently installed TopicListener.

3.69.2.4 ReturnCode_t get_qos (TopicQos *qos*) [inline]

Returns the current TopicQos settings held in the Topic **t.** This routine copies the Topic QoS properites into **qos.**

3.69.2.5 ReturnCode_t set_listener (TopicListener *new_listener,* long *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.

3.69.2.6 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_java/com/toc/coredx/DDS/Topic.java

3.70 TopicDataQosPolicy Class Reference

Allows the application to attach arbitrary information to a Topic QoS.

3.70.1 Detailed Description

Allows the application to attach arbitrary information to a Topic QoS. The value of the TOPIC_DATA is propagated and made available to applications through the built-in topics.

The topic data is implemented as a vector of Bytes, and can be used to store any application defined data.

This QoS can be used by an application in combination with the TopicListener to implement conditional matching policies.

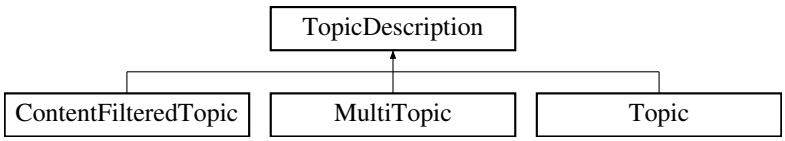
The documentation for this class was generated from the following file:

- /home/ctucker/coredx_v3.4rc/src/dds_java/com/toc/coredx/DDS/TopicDataQosPolicy.java

3.71 TopicDescription Interface Reference

TopicDescription is an abstract that provides the foundation for Topic, ContentFilteredTopic, and MultiTopic.

Inheritance diagram for TopicDescription:



Public Member Functions

- String get_type_name ()
- String get_name ()

Package Functions

- DomainParticipant get_participant ()

3.71.1 Detailed Description

TopicDescription is an abstract that provides the foundation for Topic, ContentFilteredTopic, and MultiTopic.

3.71.2 Member Function Documentation

3.71.2.1 String get_name ( )

This operation returns **topic name** of the provided TopicDescription.

Implemented in ContentFilteredTopic, MultiTopic, and Topic.

3.71.2.2 DomainParticipant get_participant ( ) [package]

This operation returns DomainParticipant that owns the TopicDescription.

Implemented in ContentFilteredTopic, MultiTopic, and Topic.

3.71.2.3 String get_type_name ( )

Gets the **type_name** of the provided TopicDescription.

Implemented in ContentFilteredTopic, MultiTopic, and Topic.

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

- /home/ctucker/coredx_v3.4rc/src/dds_java/com/toc/coredx/DDS/TopicDescription.java

3.72 TopicListener Interface Reference

The DDS_TopicListener provides asynchronous notification of DDS_Topic events. This listener can be installed during Topic creation (DDS_DomainParticipant_create_topic() and related) as well as by calling DDS_Topic_set_listener().

Public Member Functions

- long get_nil_mask ()

Package Functions

- void on_inconsistent_topic (Topic the_topic, InconsistentTopicStatus status)

3.72.1 Detailed Description

The DDS_TopicListener provides asynchronous notification of DDS_Topic events. This listener can be installed during Topic creation (DDS_DomainParticipant_create_topic() and related) as well as by calling DDS_Topic_set_listener().

3.72.2 Member Function Documentation

3.72.2.1 long get_nil_mask ( )

Returns a bitmask indicating which listener methods (if any) should be considered NIL, and therefore, should not be invoked.

3.72.2.2 void on_inconsistent_topic (Topic *the_topic*, InconsistentTopicStatus *status*) [package]

Called when the CoreDX infrastructure detects that another topic exists with different (inconsistent) characteristics.

The **status** argument provides a snapshot of the status at the time the listener was invoked.

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

- /home/ctucker/coredx_v3.4rc/src/dds_java/com/toc/coredx/DDS/TopicListener.java

3.73 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

3.73.1 Detailed Description

Structure that holds DDS_Topic Quality of Service policies.

See also

Topic::set_qos(TopicQos) set_qos()
Topic::get_qos(TopicQos) get_qos()
DomainParticipant::create_topic(java.lang.String topic_name, java.lang.String type_name, TopicQos DDS Quality of Service, TopicListener listener, long mask) create_topic()
DomainParticipant::set_default_topic_qos(TopicQos) set_default_topic_qos()

3.73.2 Member Data Documentation

3.73.2.1 DeadlineQosPolicy deadline

Defines the expected update frequency for data instances within the Topic.

3.73.2.2 DestinationOrderQosPolicy destination_order

The ordering of received samples on the Topic will be either by SOURCE or RECEPTION timestamp.

3.73.2.3 DurabilityQosPolicy durability

The durability policy of the Topic.

3.73.2.4 DurabilityServiceQosPolicy durability_service

Configures the service supporting the TRANSIENT and PERSITENT durability kinds.

3.73.2.5 HistoryQosPolicy history

The amount of historical data maintained for the Topic.

3.73.2.6 LatencyBudgetQosPolicy latency_budget

Identifies the 'urgency' of the data on the Topic. The middleware uses this to batch data samples is possible.

3.73.2.7 LifespanQosPolicy lifespan

The 'expiration time' for old data samples on the Topic.

3.73.2.8 LivelinessQosPolicy liveliness

Identifies the mechanism used to detect and maintain livelines of DataWriters on the Topic.

3.73.2.9 OwnershipQosPolicy ownership

The type of 'ownership' expected for data instances on the Topic.

3.73.2.10 ReliabilityQosPolicy reliability

Indicates the level of transport reliability on the Topic.

3.73.2.11 ResourceLimitsQosPolicy resource_limits

The resource limitations for the Topic.

3.73.2.12 TopicDataQosPolicy topic_data

A sequence of octets associated with a Topic.

3.73.2.13 TransportPriorityQosPolicy transport_priority

The priority to be used for messages on the Topic.

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

- /home/ctucker/coredx_v3.4rc/src/dds_java/com/toc/coredx/DDS/TopicQos.java

3.74 TransportPriorityQosPolicy Class Reference

A hint to the middleware to help configure the transport priority mechanism.

3.74.1 Detailed Description

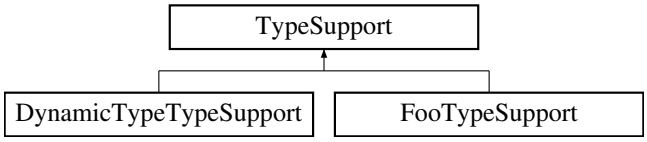
A hint to the middleware to help configure the transport priority mechanism.

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

- /home/ctucker/coredx_v3.4rc/src/dds_java/com/toc/coredx/DDS/TransportPriorityQosPolicy.java

3.75 TypeSupport Interface Reference

Inheritance diagram for TypeSupport:



Public Member Functions

- String get_type_name ()
- long getCTypeSupport ()
- DataReader create_datareader (Subscriber sub, TopicDescription td, SWIGTYPE_p__DataReader jni_dr)
- DataWriter create_datawriter (Publisher pub, Topic topic, SWIGTYPE_p__DataWriter jni_dw)

Package Functions

- ReturnCode_t register_type (DomainParticipant dp, String type_name)

3.75.1 Detailed Description

The TypeSupport interface is implemented by an application defined data type specific TypeSupport class. This provides a mechanism for an application to register an application defined data type with CoreDX DDS.

See also

com.toc.coredx.DDS.FooTypeSupport FooTypeSupport

3.75.2 Member Function Documentation

3.75.2.1 DataReader create_datareader (Subscriber *sub,* TopicDescription *td,* SWIGTYPE_p__DataReader *jni_dr*)

Private.

Implemented in DynamicTypeTypeSupport.

3.75.2.2 DataWriter create_datawriter (Publisher *pub*, Topic *topic*, SWIGTYPE_p__DataWriter *jni_dw*)

Private.

Implemented in DynamicTypeTypeSupport.

3.75.2.3 String get_type_name ( )

Returns the default name of the data type supported by this object.

Implemented in DynamicTypeTypeSupport, and FooTypeSupport.

3.75.2.4 long getCTypeSupport ( )

Private.

Implemented in DynamicTypeTypeSupport.

3.75.2.5 ReturnCode_t register_type (DomainParticipant *dp*, String *type_name*) `[package]`

Registers a new Application Defined Data Type with the DDS infrastructure. This method is implemented by the application specific TypeSupport class generated by **coredx_ddl**.

The type is registered under the name 'type_name'. If the 'type_name' parameter is null, then the default name will be used.

See also

com.toc.coredx.DDS.FooTypeSupport FooTypeSupport

Implemented in DynamicTypeTypeSupport, and FooTypeSupport.

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

- /home/ctucker/coredx_v3.4rc/src/dds_java/com/toc/coredx/DDS/TypeSupport.java

3.76 UserDataQosPolicy Class Reference

Allows the application to attach arbitrary information to a DomainParticipant, DataWriter or DataReader.

3.76.1 Detailed Description

Allows the application to attach arbitrary information to a DomainParticipant, DataWriter or DataReader. The value of the USER_DATA is propagated and made available to applications through the built-in topics.

The user data is implemented as a vector of Bytes, and can be used to store any application defined data.

This QoS can be used by an application in combination with the listeners to implement conditional matching policies.

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

- /home/ctucker/coredx_v3.4rc/src/dds_java/com/toc/coredx/DDS/UserDataQosPolicy.java

3.77 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 (Vector active_conditions, Duration_t timeout)
- ReturnCode_t get_conditions (Vector attached_conditions)

3.77.1 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

3.77.2 Constructor & Destructor Documentation

3.77.2.1 WaitSet ( ) [inline]

Constructor.

3.77.3 Member Function Documentation

3.77.3.1 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.

3.77.3.2 void destroy ( ) [inline]

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

3.77.3.3 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.

3.77.3.4 ReturnCode_t get_conditions (Vector *attached_conditions*) [inline]

Retrieves the current list of attached conditions. Populates the **attached_conditions** sequence.

3.77.3.5 ReturnCode_t wait (Vector *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_java/com/toc/coredx/DDS/WaitSet.java

3.78 WriterDataLifecycleQosPolicy Class Reference

Specifies the lifecycle behavior of data instances managed by the DataWriter. If autodispose_unregistered_-instances is true, then the DataWriter will automatically dispose any instances that are unregistered. **Note:** When a DataWriter is deleted, it will automatically unregister all of its instances. With this policy == true, then all instances will also be disposed.

3.78.1 Detailed Description

Specifies the lifecycle behavior of data instances managed by the DataWriter. If autodispose_unregistered_-instances is true, then the DataWriter will automatically dispose any instances that are unregistered. **Note:** When a DataWriter is deleted, it will automatically unregister all of its instances. With this policy == true, then all instances will also be disposed.

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

- /home/ctucker/coredx_v3.4rc/src/dds_java/com/toc/coredx/DDS/WriterDataLifecycleQosPolicy.java

Chapter 4

Data Structure Index

4.1 Class List

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

ArrayDynamicType (ArrayDynamicType holds an 'Array' value) . . . . . ??
BooleanDynamicType (BooleanDynamicType holds an 'boolean' value) . . . . . ??
CharDynamicType (CharDynamicType holds an 'char' value) . . . . . ??
Condition (A Condition can be added to a DDS_WaitSet to provide synchronous event notification) 15
ContentFilteredTopic (ContentFilteredTopic provides a topic that may include data filtered from a related Topic) 17
DataReader (The DataReader entity allows the application to subscribe to and read data) 19
DataReaderListener (The DataReaderListener provides asynchronous notification of DataReader events. This listener can be installed during DataReader creation, DomainParticipant_-create_yyy(), as well as by calling DataReader_set_listener()) 25
DataReaderQos (Structure that holds DataReader Quality of Service policies) 27
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) 30
DataWriterListener (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()) 34
DataWriterQos (Structure that holds DataWriter Quality of Service policies) 36
DDS 39
DeadlineQosPolicy (This QoS policy establishes a minimum update period for data instances) . . 40
DestinationOrderQosPolicy (This QoS policy controls how each Subscriber orders received data samples) 41
DomainEntity (Base class for all DDS Domain Entities) 42
DomainId_t 43

DomainParticipant (The DomainParticipant is used to configure, create and destroy Publisher, Subscriber and Topic objects) . . . 44
DomainParticipantFactory (DomainParticipantFactory constructs DomainParticipants. The) . . . 54
DomainParticipantFactoryQos (Structure that holds DomainParticipantFactory Quality of Service policies) . . . 57
DomainParticipantListener . . . 58
DomainParticipantQos (Structure that holds DomainParticipant Quality of Service policies) . . . 62
DoubleDynamicType (DoubleDynamicType holds an 'double' value) . . . **??**
DurabilityQosPolicy . . . 63
DurabilityServiceQosPolicy . . . 64
Duration_t (Represents a time duration with nanosecond resolution) . . . 65
DynamicType (DynamicType is the base class for all specific DynamicTypes) . . . **??**
DynamicTypeDataReader (DynamicTypeDataReader is an implementation of DataReader that supports DynamicTypes) . . . **??**
DynamicTypeDataWriter (DynamicTypeDataWriter is an implementation of DataWriter that supports DynamicTypes) . . . **??**
DynamicTypeTypeSupport (DynamicTypeTypeSupport is an implementation of TypeSupport that supports DynamicTypes) . . . **??**
Entity (Base class for all DDS Entities) . . . 66
EntityFactoryQosPolicy . . . 68
EntityNameQosPolicy . . . 69
FloatDynamicType (FloatDynamicType holds an 'float' value) . . . **??**
FooDataReader (The FooDataReader is an example specialized DataReader (generated by the coredx_ddl compiler) for reading data samples of type 'Foo') . . . 70
FooDataWriter (The FooDataWriter is an example specialized DataWriter (generated by the coredx_ddl compiler) for writing data samples of type 'Foo') . . . 78
FooTypeSupport (Impliments the TypeSupport inteface for the Data Type 'Foo') . . . 81
GroupDataQosPolicy (Allows the application to attach arbitrary information to a Publisher or Subscriber) . . . 83
GuardCondition (A GuardCondition is a Condition where the **trigger_value** is under application control) . . . 84
HistoryQosPolicy (Controls the ammount of historical data maintained by a DataReader or DataWriter) . . . 85
InconsistentTopicStatus (InconsistentTopicStatus provides status related to the on_inconsistent_topic listener methods of the TopicListener structure) . . . 86
InstanceHandle_t (Used to identify a DDS Entity or data instance. An InstanceHandle is unique within its context. For example, each data instance known by a DataReader is assigned a unique InstanceHandle. However, these handles may not be unique when compared to handles from another DataReader) . . . 87
LatencyBudgetQosPolicy (Specifies allowable latency) . . . 88
LifespanQosPolicy (Specifies the maximum duration of validity of the data written by the DataWriter) . . . 89
LivelinessChangedStatus (Status related to the on_liveliness_changed listener methods of the DataReader, Subscriber, and DomainParticipant structures) . . . 90
LivelinessLostStatus (Status related to the on_liveliness_lost listener methods of the DataWriter, Publisher, and DomainParticipant structures) . . . 92

LivelinessQosPolicy (Determines the mechanism and parameters used by the application to determine whether an Entity is alive) 93
LongDynamicType (LongDynamicType holds an 'long' value (32bits, signed)) . . . . . ??
LongLongDynamicType (LongLongDynamicType holds an 'long long' (64bit, signed) value) . . ??
MultiTopic (MultiTopic provides a topic that may include data from multiple Topics) 94
OctetDynamicType (OctetDynamicType holds an 'octet' value) . . . . . ??
OfferedDeadlineMissedStatus (Status related to the on_offered_deadline_missed listener methods of the DataWriter, Publisher, and DomainParticipant structures) 96
OfferedIncompatibleQosStatus (Status related to the on_offered_incompatible_qos listener methods of the DataWriter, Publisher, and DomainParticipant structures) 97
OwnershipQosPolicy (Determines instance ownership in the case of multple writers. CoreDX DDS supports both SHARED_OWNERSHIP_QOS and EXCLUSIVE_OWNERSHIP_QOS) 98
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) 99
ParticipantBuiltinTopicData (The ParticipantBuiltinTopicData is a built-in Data Type that represents Participant Discovery information) . . . . . ??
ParticipantBuiltinTopicDataDataReader (The ParticipantBuiltinTopicDataDataReader is a built-in DataReader that can be used to access Participant Discovery information. The reader can be accessed through the special 'builtin' Subscriber via the DomainParticipant.get_builtin_subscriber() routine) 100
PartitionQosPolicy 101
PresentationQosPolicy (Controls the presentation of received data samples to the application. CoreDX DDS currently supports only the access_scope = INSTANCE_PRESENTATION_QOS policy) 102
PublicationBuiltinTopicData (The PublicationBuiltinTopicData is a built-in Data Type that represents Publication Discovery information) . . . . . ??
PublicationBuiltinTopicDataDataReader (The PublicationBuiltinTopicDataDataReader is a built-in DataReader that can be used to access Publication Discovery information. The reader can be accessed through the special 'builtin' Subscriber via the DomainParticipant.get_builtin_subscriber() routine) 103
PublicationMatchedStatus (Status related to the on_publication_matched listener methods of the DataWriter, Publisher, and DomainParticipant structures) 104
Publisher (The Publisher configures, creates, manages and destroys DataWriters) 106
PublisherListener 111
PublisherQos (Structure that holds Publisher Quality of Service policies) 113
QueryCondition (The **trigger_value** is driven by the data available, after applying the filter, in the associated DataReader) 115
ReadCondition (A ReadCondition is a specialized Condition associated with a DataReader) . . . 117
ReaderDataLifecycleQosPolicy (Specifies the lifecycle behavior of data instances managed by the DataReader) 119
ReliabilityQosPolicy (Indicates the level of reliability offered/provided by the Entity. If kind is RELIABLE_RELIABILITY_QOS, then the middleware will attempt to deliver all samples in the history cache. If samples are not received, then they will be retried) 120

RequestedDeadlineMissedStatus (Status related to the on_requested_deadline_missed listener methods of the DataReader, Subscriber, and DomainParticipant structures) 121
RequestedIncompatibleQosStatus 122
ResourceLimitsQosPolicy (Specifies the resources that the Service can use to maintain data samples and instances) 123
SampleInfo (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) 124
SampleLostStatus (Status related to the on_sample_lost listener methods of the DataReader, Subscriber, and DomainParticipant structures) 127
SampleRejectedStatus (Status related to the on_sample_rejected listener methods of the DataReader, Subscriber, and DomainParticipant structures) 128
SequenceDynamicType (SequenceDynamicType holds an 'Sequence' value) . . . . . **??**
ShortDynamicType (ShortDynamicType holds an 'short' value) . . . . . **??**
StatusCondition (A StatusCondition is a condition associated with an Entity. The **trigger_value** is driven by the communication status of the associated Entity) 129
StringDynamicType (StringDynamicType holds an 'string' value) . . . . . **??**
StructDynamicType (StructDynamicType represents structures) . . . . . **??**
Subscriber (The Subscriber configures, creates, manages and destroys DataReaders) 131
SubscriberListener (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()) 136
SubscriberQos (Structure that holds DDS_Subscriber Quality of Service policies) 139
SubscriptionBuiltinTopicData (The SubscriptionBuiltinTopicData is a built-in Data Type that represents Subscription Discovery information) . . . . . **??**
SubscriptionBuiltinTopicDataDataReader (The SubscriptionBuiltinTopicDataDataReader is a built-in DataReader that can be used to access Subscription Discovery information) . . . 141
SubscriptionMatchedStatus (Status related to the on_subscription_matched listener methods of the DataReader, Subscriber, and DomainParticipant structures) 142
Time_t (Represents time with nanosecond resolution) 144
TimeBasedFilterQosPolicy (Defines a filter based on time between samples. The DataReader indicates that it wants at most one sample for each instance every minimum_separation interval) 145
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.]) 146
TopicDataQosPolicy (Allows the application to attach arbitrary information to a Topic QoS) . . . 149
TopicDescription (TopicDescription is an abstract that provides the foundation for Topic, ContentFilteredTopic, and MultiTopic) 150
TopicListener (The DDS_TopicListener provides asynchronous notification of DDS_Topic events. This listener can be installed during Topic creation (DDS_DomainParticipant_create_-topic() and related) as well as by calling DDS_Topic_set_listener()) 152

TopicQos (Structure that holds DDS_Topic Quality of Service policies) 153
TransportPriorityQosPolicy (A hint to the middleware to help configure the transport priority mechanism) 156
TypeSupport 157
ULongDynamicType (ULongDynamicType holds an 'unsigned long' value (32bits, unsigned)) . . ??
ULongLongDynamicType (ULongLongDynamicType holds an 'long long' (64bit, unsigned) value) ??
UnionDynamicType (UnionDynamicType represents unionures) . . . . . ??
UserDataQosPolicy (Allows the application to attach arbitrary information to a DomainParticipant, DataWriter or DataReader) 159
UShortDynamicType (UShortDynamicType holds an 'unsigned short' value) . . . . . ??
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) 160
WriterDataLifecycleQosPolicy (Specifies the lifecycle behavior of data instances managed by the DataWriter. If autodispose_unregistered_instances is true, then the DataWriter will automatically dispose any instances that are unregistered. **Note:** When a DataWriter is deleted, it will automatically unregister all of its instances. With this policy == true, then all instances will also be disposed) 162

Chapter 5

Not Yet Supported

Member DataReader::create_querycondition(long sample_states, long view_states, long instance_states, String query_
QueryConditions are not yet supported as triggers for a WaitSet.

Member DomainParticipant::create_multitopic(String name, String type_name, String subscription_expression, Vecto
This is currently unsupported in the Java language binding.

Member DomainParticipant::delete_multitopic(MultiTopic a_multitopic) This is currently unsupported in the Java language binding.

Member DomainParticipant::get_discovered_topic_data(TopicBuiltinTopicData topic_data, InstanceHandle_t topic_ha
This is currently unsupported in the Java language binding.

Member DomainParticipant::get_discovered_topics(Vector topic_handles) This is currently unsupported in the Java language binding.

Member DomainParticipant::ignore_participant(InstanceHandle_t handle) This is currently unsupported in the Java language binding.

Member DomainParticipant::ignore_publication(InstanceHandle_t handle) This is currently unsupported in the Java language binding.

Member DomainParticipant::ignore_subscription(InstanceHandle_t handle) This is currently unsupported in the Java language binding.

Member DomainParticipant::ignore_topic(InstanceHandle_t handle) This is currently unsupported in the Java language binding.

Class MultiTopic This is currently unsupported in the Java language binding.

Member MultiTopic::get_name() This routine is not yet implemented.

Member MultiTopic::get_participant() This routine is not yet implemented.

Member MultiTopic::get_type_name() This routine is not yet implemented.

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 Java language binding.

Member Subscriber::end_access() This is currently unsupported in the Java language binding.

Member Subscriber::get_datareaders(Vector readers, long sample_states, long view_states, long instance_states) This is currently unsupported in the Java language binding.

Index

%DDS Status Structures, 12

absolute_generation_rank
    com::toc::coredx::DDS::SampleInfo, 124
assert_liveliness
    com::toc::coredx::DDS::DataWriter, 31
    com::toc::coredx::DDS::DomainParticipant, 45
attach_condition
    com::toc::coredx::DDS::WaitSet, 160

begin_access
    com::toc::coredx::DDS::Subscriber, 132
begin_coherent_changes
    com::toc::coredx::DDS::Publisher, 107

com::toc::coredx::DDS::Condition, 15
    get_trigger_value, 16
com::toc::coredx::DDS::ContentFilteredTopic, 17
    get_expression_parameters, 17
    get_related_topic, 18
    set_expression_parameters, 18
com::toc::coredx::DDS::DataReader, 19
    create_querycondition, 20
    create_readcondition, 20
    delete_contained_entities, 21
    delete_readcondition, 21
    enable, 21
    get_listener, 21
    get_liveliness_changed_status, 22
    get_matched_publication_data, 22
    get_matched_publications, 22
    get_qos, 22
    get_requested_deadline_missed_status, 22
    get_requested_incompatible_qos_status, 22
    get_sample_lost_status, 23
    get_sample_rejected_status, 23
    get_subscriber, 23
    get_subscription_matched_status, 23
    get_topicdescription, 23
    set_listener, 23
    set_qos, 23
    wait_for_historical_data, 24
com::toc::coredx::DDS::DataReaderListener, 25
    get_nil_mask, 25
    on_data_available, 25
    on_liveliness_changed, 25
    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, 27
    deadline, 27
    destination_order, 27
    durability, 27
    history, 28
    latency_budget, 28
    liveliness, 28
    ownership, 28
    reader_data_lifecycle, 28
    reliability, 28
    resource_limits, 28
    time_based_filter, 28
    user_data, 28
com::toc::coredx::DDS::DataWriter, 30
    assert_liveliness, 31
    enable, 31
    get_listener, 31
    get_liveliness_lost_status, 31
    get_matched_subscription_data, 31
    get_matched_subscriptions, 32
    get_offered_deadline_missed_status, 32
    get_offered_incompatible_qos_status, 32
    get_publication_matched_status, 32

get_publisher, 32
get_qos, 32
get_topic, 33
set_listener, 33
set_qos, 33
wait_for_acknowledgments, 33
com::toc::coredx::DDS::DataWriterListener, 34
get_nil_mask, 34
on_liveliness_lost, 34
on_offered_deadline_missed, 34
on_offered_incompatible_qos, 35
on_publication_matched, 35
com::toc::coredx::DDS::DataWriterQos, 36
deadline, 36
destination_order, 36
durability, 37
durability_service, 37
history, 37
latency_budget, 37
lifespan, 37
liveliness, 37
ownership, 37
ownership_strength, 37
reliability, 37
resource_limits, 37
transport_priority, 38
user_data, 38
writer_data_lifecycle, 38
com::toc::coredx::DDS::DDS, 39
error_str, 39
qos_policy_str, 39
com::toc::coredx::DDS::DeadlineQosPolicy, 40
com::toc::coredx::DDS::DestinationOrderQosPolicy, 41
com::toc::coredx::DDS::DomainEntity, 42
com::toc::coredx::DDS::DomainId_t, 43
com::toc::coredx::DDS::DomainParticipant, 44
assert_liveliness, 45
contains_entity, 45
create_contentfilteredtopic, 46
create_multitopic, 46
create_publisher, 46
create_subscriber, 46
create_topic, 47
delete_contained_entities, 47
delete_contentfilteredtopic, 47
delete_multitopic, 47
delete_publisher, 48
delete_subscriber, 48
delete_topic, 48
enable, 48
find_topic, 48
get_builtin_subscriber, 49
get_current_time, 49
get_default_publisher_qos, 49
get_default_subscriber_qos, 49
get_default_topic_qos, 49
get_discovered_participant_data, 50
get_discovered_participants, 50
get_discovered_topic_data, 50
get_discovered_topics, 50
get_domain_id, 50
get_listener, 51
get_qos, 51
ignore_participant, 51
ignore_publication, 51
ignore_subscription, 51
ignore_topic, 52
lookup_topicdescription, 52
register_type, 52
set_default_publisher_qos, 52
set_default_subscriber_qos, 52
set_default_topic_qos, 53
set_listener, 53
set_qos, 53
com::toc::coredx::DDS::DomainParticipantFactory, 54
create_participant, 54
delete_participant, 55
get_default_participant_qos, 55
get_instance, 55
get_qos, 55
lookup_participant, 55
set_default_participant_qos, 55
set_qos, 56
com::toc::coredx::DDS::DomainParticipantFactoryQos, 57
entity_factory, 57
com::toc::coredx::DDS::DomainParticipantListener, 58
get_nil_mask, 58
on_data_available, 58

on_data_on_readers, 58
on_inconsistent_topic, 59
on_liveliness_changed, 59
on_liveliness_lost, 59
on_offered_deadline_missed, 59
on_offered_incompatible_qos, 59
on_publication_matched, 60
on_requested_deadline_missed, 60
on_requested_incompatible_qos, 60
on_sample_lost, 60
on_sample_rejected, 61
on_subscription_matched, 61
com::toc::coredx::DDS::DomainParticipantQos, 62
entity_factory, 62
peer_participants, 62
user_data, 62
com::toc::coredx::DDS::DurabilityQosPolicy, 63
com::toc::coredx::DDS::DurabilityServiceQosPolicy, 64
com::toc::coredx::DDS::Duration_t, 65
com::toc::coredx::DDS::Entity, 66
enable, 66
get_instance_handle, 66
get_status_changes, 66
get_statuscondition, 66
com::toc::coredx::DDS::EntityFactoryQosPolicy, 68
com::toc::coredx::DDS::EntityNameQosPolicy, 69
com::toc::coredx::DDS::FooDataReader, 70
get_key_value, 71
lookup_instance, 71
read, 71
read_instance, 72
read_next_instance, 72
read_next_instance_w_condition, 73
read_next_sample, 73
read_w_condition, 73
return_loan, 74
take, 74
take_instance, 75
take_next_instance, 76
take_next_instance_w_condition, 76
take_next_sample, 76
take_w_condition, 76
com::toc::coredx::DDS::FooDataWriter, 78
dispose, 79
dispose_w_timestamp, 79
get_key_value, 79
lookup_instance, 79
register_instance, 79
register_instance_w_timestamp, 79
unregister_instance, 79
unregister_instance_w_timestamp, 80
write, 80
write_w_timestamp, 80
com::toc::coredx::DDS::FooTypeSupport, 81
get_type_name, 81
register_type, 81
com::toc::coredx::DDS::GroupDataQosPolicy, 83
com::toc::coredx::DDS::GuardCondition, 84
GuardCondition, 84
com::toc::coredx::DDS::HistoryQosPolicy, 85
com::toc::coredx::DDS::InconsistentTopicStatus, 86
get_total_count, 86
get_total_count_change, 86
com::toc::coredx::DDS::InstanceHandle_t, 87
com::toc::coredx::DDS::LatencyBudgetQosPolicy, 88
com::toc::coredx::DDS::LifespanQosPolicy, 89
com::toc::coredx::DDS::LivelinessChangedStatus, 90
get_alive_count, 90
get_alive_count_change, 90
get_last_publication_handle, 90
get_not_alive_count, 90
get_not_alive_count_change, 90
com::toc::coredx::DDS::LivelinessLostStatus, 92
get_total_count, 92
get_total_count_change, 92
com::toc::coredx::DDS::LivelinessQosPolicy, 93
com::toc::coredx::DDS::MultiTopic, 94
get_name, 94
get_participant, 94
get_type_name, 94
com::toc::coredx::DDS::OfferedDeadlineMissedStatus, 96
get_last_instance_handle, 96
get_total_count, 96
get_total_count_change, 96
com::toc::coredx::DDS::OfferedIncompatibleQosStatus, 97
get_last_policy_id, 97
get_policies, 97

get_total_count, 97
get_total_count_change, 97
com::toc::coredx::DDS::OwnershipQosPolicy, 98
com::toc::coredx::DDS::OwnershipStrengthQosPolicy, 99
com::toc::coredx::DDS::ParticipantBuiltinTopicDataDataReader, 100
com::toc::coredx::DDS::PartitionQosPolicy, 101
com::toc::coredx::DDS::PresentationQosPolicy, 102
com::toc::coredx::DDS::PublicationBuiltinTopicDataDataReader, 103
com::toc::coredx::DDS::PublicationMatchedStatus, 104
get_current_count, 104
get_current_count_change, 104
get_last_subscription_handle, 104
get_total_count, 104
get_total_count_change, 104
com::toc::coredx::DDS::Publisher, 106
begin_coherent_changes, 107
copy_from_topic_qos, 107
create_datawriter, 107
delete_contained_entities, 107
delete_datawriter, 107
enable, 107
end_coherent_changes, 108
get_default_datawriter_qos, 108
get_listener, 108
get_participant, 108
get_qos, 108
lookup_datawriter, 108
resume_publications, 109
set_default_datawriter_qos, 109
set_listener, 109
set_qos, 109
suspend_publications, 109
wait_for_acknowledgments, 109
com::toc::coredx::DDS::PublisherListener, 111
get_nil_mask, 111
on_liveliness_lost, 111
on_offered_deadline_missed, 111
on_offered_incompatible_qos, 111
on_publication_matched, 112
com::toc::coredx::DDS::PublisherQos, 113
entity_factory, 113
group_data, 113
partition, 113
presentation, 113
com::toc::coredx::DDS::QueryCondition, 115
get_query_expression, 116
get_query_parameters, 116
set_query_parameters, 116
com::toc::coredx::DDS::ReadCondition, 117
get_datareader, 117
get_instance_state_mask, 117
get_sample_state_mask, 117
get_view_state_mask, 118
com::toc::coredx::DDS::ReaderDataLifecycleQosPolicy, 119
com::toc::coredx::DDS::ReliabilityQosPolicy, 120
com::toc::coredx::DDS::RequestedDeadlineMissedStatus, 121
get_last_instance_handle, 121
get_total_count, 121
get_total_count_change, 121
com::toc::coredx::DDS::RequestedIncompatibleQosStatus, 122
get_last_policy_id, 122
get_policies, 122
get_total_count, 122
get_total_count_change, 122
com::toc::coredx::DDS::ResourceLimitsQosPolicy, 123
com::toc::coredx::DDS::SampleInfo, 124
absolute_generation_rank, 124
disposed_generation_count, 124
generation_rank, 124
instance_handle, 125
instance_state, 125
no_writers_generation_count, 125
publication_handle, 125
sample_rank, 125
sample_state, 125
source_timestamp, 126
valid_data, 126
view_state, 126
com::toc::coredx::DDS::SampleLostStatus, 127
get_total_count, 127
get_total_count_change, 127
com::toc::coredx::DDS::SampleRejectedStatus, 128
get_last_instance_handle, 128
get_last_reason, 128

get_total_count, 128
get_total_count_change, 128
com::toc::coredx::DDS::StatusCondition, 129
get_enabled_statuses, 129
get_entity, 129
set_enabled_statuses, 129
com::toc::coredx::DDS::Subscriber, 131
begin_access, 132
copy_from_topic_qos, 132
create_datareader, 132
delete_contained_entities, 132
delete_datareader, 132
enable, 133
end_access, 133
get_datareaders, 133
get_default_datareader_qos, 134
get_listener, 134
get_participant, 134
get_qos, 134
lookup_datareader, 134
set_default_datareader_qos, 134
set_listener, 135
set_qos, 135
com::toc::coredx::DDS::SubscriberListener, 136
get_nil_mask, 136
on_data_available, 136
on_data_on_readers, 136
on_liveliness_changed, 137
on_requested_deadline_missed, 137
on_requested_incompatible_qos, 137
on_sample_lost, 137
on_sample_rejected, 137
on_subscription_matched, 138
com::toc::coredx::DDS::SubscriberQos, 139
entity_factory, 139
group_data, 139
partition, 139
presentation, 139
com::toc::coredx::DDS::SubscriptionBuiltinTopicDataDataReader, 141
com::toc::coredx::DDS::SubscriptionMatchedStatus, 142
get_current_count, 142
get_current_count_change, 142
get_last_publication_handle, 142
get_total_count, 142
get_total_count_change, 142
com::toc::coredx::DDS::Time_t, 144
com::toc::coredx::DDS::TimeBasedFilterQosPolicy, 145
com::toc::coredx::DDS::Topic, 146
enable, 147
get_inconsistent_topic_status, 147
get_listener, 147
get_qos, 147
set_listener, 147
set_qos, 147
com::toc::coredx::DDS::TopicDataQosPolicy, 149
com::toc::coredx::DDS::TopicDescription, 150
get_name, 150
get_participant, 150
get_type_name, 150
com::toc::coredx::DDS::TopicListener, 152
get_nil_mask, 152
on_inconsistent_topic, 152
com::toc::coredx::DDS::TopicQos, 153
deadline, 153
destination_order, 153
durability, 153
durability_service, 154
history, 154
latency_budget, 154
lifespan, 154
liveliness, 154
ownership, 154
reliability, 154
resource_limits, 154
topic_data, 154
transport_priority, 154
com::toc::coredx::DDS::TransportPriorityQosPolicy, 156
com::toc::coredx::DDS::TypeSupport, 157
create_datareader, 157
create_datawriter, 157
get_type_name, 158
getCTypeSupport, 158
register_type, 158
com::toc::coredx::DDS::UserDataQosPolicy, 159
com::toc::coredx::DDS::WaitSet, 160
attach_condition, 160
destroy, 160
detach_condition, 160

get_conditions, 161
wait, 161
WaitSet, 160
com::toc::coredx::DDS::WriterDataLifecycleQosPolicy, 162
contains_entity
com::toc::coredx::DDS::DomainParticipant, 45
copy_from_topic_qos
com::toc::coredx::DDS::Publisher, 107
com::toc::coredx::DDS::Subscriber, 132
create_contentfilteredtopic
com::toc::coredx::DDS::DomainParticipant, 46
create_datareader
com::toc::coredx::DDS::Subscriber, 132
com::toc::coredx::DDS::TypeSupport, 157
create_datawriter
com::toc::coredx::DDS::Publisher, 107
com::toc::coredx::DDS::TypeSupport, 157
create_multitopic
com::toc::coredx::DDS::DomainParticipant, 46
create_participant
com::toc::coredx::DDS::DomainParticipantFactory, 54
create_publisher
com::toc::coredx::DDS::DomainParticipant, 46
create_querycondition
com::toc::coredx::DDS::DataReader, 20
create_readcondition
com::toc::coredx::DDS::DataReader, 20
create_subscriber
com::toc::coredx::DDS::DomainParticipant, 46
create_topic
com::toc::coredx::DDS::DomainParticipant, 47

DDS Conditions, 10
DDS Conditions, Listeners, and WaitSets, 8
DDS Entities, 5
DDS Listeners, 9
DDS Quality of Service, 7
DDS WaitSets, 11
deadline
com::toc::coredx::DDS::DataReaderQos, 27
com::toc::coredx::DDS::DataWriterQos, 36
com::toc::coredx::DDS::TopicQos, 153
delete_contained_entities
com::toc::coredx::DDS::DataReader, 21
com::toc::coredx::DDS::DomainParticipant, 47
com::toc::coredx::DDS::Publisher, 107
com::toc::coredx::DDS::Subscriber, 132
delete_contentfilteredtopic
com::toc::coredx::DDS::DomainParticipant, 47
delete_datareader
com::toc::coredx::DDS::Subscriber, 132
delete_datawriter
com::toc::coredx::DDS::Publisher, 107
delete_multitopic
com::toc::coredx::DDS::DomainParticipant, 47
delete_participant
com::toc::coredx::DDS::DomainParticipantFactory, 55
delete_publisher
com::toc::coredx::DDS::DomainParticipant, 48
delete_readcondition
com::toc::coredx::DDS::DataReader, 21
delete_subscriber
com::toc::coredx::DDS::DomainParticipant, 48
delete_topic
com::toc::coredx::DDS::DomainParticipant, 48
destination_order
com::toc::coredx::DDS::DataReaderQos, 27
com::toc::coredx::DDS::DataWriterQos, 36
com::toc::coredx::DDS::TopicQos, 153
destroy
com::toc::coredx::DDS::WaitSet, 160
detach_condition
com::toc::coredx::DDS::WaitSet, 160
dispose
com::toc::coredx::DDS::FooDataWriter, 79
dispose_w_timestamp
com::toc::coredx::DDS::FooDataWriter, 79
disposed_generation_count
com::toc::coredx::DDS::SampleInfo, 124
durability
com::toc::coredx::DDS::DataReaderQos, 27
com::toc::coredx::DDS::DataWriterQos, 37
com::toc::coredx::DDS::TopicQos, 153
durability_service
com::toc::coredx::DDS::DataWriterQos, 37
com::toc::coredx::DDS::TopicQos, 154

enable
com::toc::coredx::DDS::DataReader, 21

com::toc::coredx::DDS::DataWriter, 31
com::toc::coredx::DDS::DomainParticipant, 48
com::toc::coredx::DDS::Entity, 66
com::toc::coredx::DDS::Publisher, 107
com::toc::coredx::DDS::Subscriber, 133
com::toc::coredx::DDS::Topic, 147
end_access
com::toc::coredx::DDS::Subscriber, 133
end_coherent_changes
com::toc::coredx::DDS::Publisher, 108
entity_factory
com::toc::coredx::DDS::DomainParticipantFactoryQos, 57
com::toc::coredx::DDS::DomainParticipantQos, 62
com::toc::coredx::DDS::PublisherQos, 113
com::toc::coredx::DDS::SubscriberQos, 139
error_str
com::toc::coredx::DDS::DDS, 39

find_topic
com::toc::coredx::DDS::DomainParticipant, 48

generation_rank
com::toc::coredx::DDS::SampleInfo, 124
get_alive_count
com::toc::coredx::DDS::LivelinessChangedStatus, 90
get_alive_count_change
com::toc::coredx::DDS::LivelinessChangedStatus, 90
get_builtin_subscriber
com::toc::coredx::DDS::DomainParticipant, 49
get_conditions
com::toc::coredx::DDS::WaitSet, 161
get_current_count
com::toc::coredx::DDS::PublicationMatchedStatus, 104
com::toc::coredx::DDS::SubscriptionMatchedStatus, 142
get_current_count_change
com::toc::coredx::DDS::PublicationMatchedStatus, 104
com::toc::coredx::DDS::SubscriptionMatchedStatus, 142
get_current_time
com::toc::coredx::DDS::DomainParticipant, 49
get_datareader
com::toc::coredx::DDS::ReadCondition, 117
get_datareaders
com::toc::coredx::DDS::Subscriber, 133
get_default_datareader_qos
com::toc::coredx::DDS::Subscriber, 134
get_default_datawriter_qos
com::toc::coredx::DDS::Publisher, 108
get_default_participant_qos
com::toc::coredx::DDS::DomainParticipantFactory, 55
get_default_publisher_qos
com::toc::coredx::DDS::DomainParticipant, 49
get_default_subscriber_qos
com::toc::coredx::DDS::DomainParticipant, 49
get_default_topic_qos
com::toc::coredx::DDS::DomainParticipant, 49
get_discovered_participant_data
com::toc::coredx::DDS::DomainParticipant, 50
get_discovered_participants
com::toc::coredx::DDS::DomainParticipant, 50
get_discovered_topic_data
com::toc::coredx::DDS::DomainParticipant, 50
get_discovered_topics
com::toc::coredx::DDS::DomainParticipant, 50
get_domain_id
com::toc::coredx::DDS::DomainParticipant, 50
get_enabled_statuses
com::toc::coredx::DDS::StatusCondition, 129
get_entity
com::toc::coredx::DDS::StatusCondition, 129
get_expression_parameters
com::toc::coredx::DDS::ContentFilteredTopic, 17
get_inconsistent_topic_status
com::toc::coredx::DDS::Topic, 147
get_instance
com::toc::coredx::DDS::DomainParticipantFactory, 55
get_instance_handle
com::toc::coredx::DDS::Entity, 66
get_instance_state_mask
com::toc::coredx::DDS::ReadCondition, 117
get_key_value
com::toc::coredx::DDS::FooDataReader, 71

com::toc::coredx::DDS::FooDataWriter, 79
get_last_instance_handle
com::toc::coredx::DDS::OfferedDeadlineMissedStatus, 96
com::toc::coredx::DDS::RequestedDeadlineMissedStatus, 121
com::toc::coredx::DDS::SampleRejectedStatus, 128
get_last_policy_id
com::toc::coredx::DDS::OfferedIncompatibleQosStatus, 97
com::toc::coredx::DDS::RequestedIncompatibleQosStatus, 122
get_last_publication_handle
com::toc::coredx::DDS::LivelinessChangedStatus, 90
com::toc::coredx::DDS::SubscriptionMatchedStatus, 142
get_last_reason
com::toc::coredx::DDS::SampleRejectedStatus, 128
get_last_subscription_handle
com::toc::coredx::DDS::PublicationMatchedStatus, 104
get_listener
com::toc::coredx::DDS::DataReader, 21
com::toc::coredx::DDS::DataWriter, 31
com::toc::coredx::DDS::DomainParticipant, 51
com::toc::coredx::DDS::Publisher, 108
com::toc::coredx::DDS::Subscriber, 134
com::toc::coredx::DDS::Topic, 147
get_liveliness_changed_status
com::toc::coredx::DDS::DataReader, 22
get_liveliness_lost_status
com::toc::coredx::DDS::DataWriter, 31
get_matched_publication_data
com::toc::coredx::DDS::DataReader, 22
get_matched_publications
com::toc::coredx::DDS::DataReader, 22
get_matched_subscription_data
com::toc::coredx::DDS::DataWriter, 31
get_matched_subscriptions
com::toc::coredx::DDS::DataWriter, 32
get_name
com::toc::coredx::DDS::MultiTopic, 94
com::toc::coredx::DDS::TopicDescription, 150
get_nil_mask
com::toc::coredx::DDS::DataReaderListener, 25
com::toc::coredx::DDS::DataWriterListener, 34
com::toc::coredx::DDS::DomainParticipantListener, 58
com::toc::coredx::DDS::PublisherListener, 111
com::toc::coredx::DDS::SubscriberListener, 136
com::toc::coredx::DDS::TopicListener, 152
get_not_alive_count
com::toc::coredx::DDS::LivelinessChangedStatus, 90
get_not_alive_count_change
com::toc::coredx::DDS::LivelinessChangedStatus, 90
get_offered_deadline_missed_status
com::toc::coredx::DDS::DataWriter, 32
get_offered_incompatible_qos_status
com::toc::coredx::DDS::DataWriter, 32
get_participant
com::toc::coredx::DDS::MultiTopic, 94
com::toc::coredx::DDS::Publisher, 108
com::toc::coredx::DDS::Subscriber, 134
com::toc::coredx::DDS::TopicDescription, 150
get_policies
com::toc::coredx::DDS::OfferedIncompatibleQosStatus, 97
com::toc::coredx::DDS::RequestedIncompatibleQosStatus, 122
get_publication_matched_status
com::toc::coredx::DDS::DataWriter, 32
get_publisher
com::toc::coredx::DDS::DataWriter, 32
get_qos
com::toc::coredx::DDS::DataReader, 22
com::toc::coredx::DDS::DataWriter, 32
com::toc::coredx::DDS::DomainParticipant, 51
com::toc::coredx::DDS::DomainParticipantFactory, 55
com::toc::coredx::DDS::Publisher, 108
com::toc::coredx::DDS::Subscriber, 134
com::toc::coredx::DDS::Topic, 147
get_query_expression
com::toc::coredx::DDS::QueryCondition, 116
get_query_parameters

com::toc::coredx::DDS::QueryCondition, 116
get_related_topic
com::toc::coredx::DDS::ContentFilteredTopic, 18
get_requested_deadline_missed_status
com::toc::coredx::DDS::DataReader, 22
get_requested_incompatible_qos_status
com::toc::coredx::DDS::DataReader, 22
get_sample_lost_status
com::toc::coredx::DDS::DataReader, 23
get_sample_rejected_status
com::toc::coredx::DDS::DataReader, 23
get_sample_state_mask
com::toc::coredx::DDS::ReadCondition, 117
get_status_changes
com::toc::coredx::DDS::Entity, 66
get_statuscondition
com::toc::coredx::DDS::Entity, 66
get_subscriber
com::toc::coredx::DDS::DataReader, 23
get_subscription_matched_status
com::toc::coredx::DDS::DataReader, 23
get_topic
com::toc::coredx::DDS::DataWriter, 33
get_topicdescription
com::toc::coredx::DDS::DataReader, 23
get_total_count
com::toc::coredx::DDS::InconsistentTopicStatus, 86
com::toc::coredx::DDS::LivelinessLostStatus, 92
com::toc::coredx::DDS::OfferedDeadlineMissedStatus, 96
com::toc::coredx::DDS::OfferedIncompatibleQosStatus, 97
com::toc::coredx::DDS::PublicationMatchedStatus, 104
com::toc::coredx::DDS::RequestedDeadlineMissedStatus, 121
com::toc::coredx::DDS::RequestedIncompatibleQosStatus, 122
com::toc::coredx::DDS::SampleLostStatus, 127
com::toc::coredx::DDS::SampleRejectedStatus, 128
com::toc::coredx::DDS::SubscriptionMatchedStatus, 142
get_total_count_change
com::toc::coredx::DDS::InconsistentTopicStatus, 86
com::toc::coredx::DDS::LivelinessLostStatus, 92
com::toc::coredx::DDS::OfferedDeadlineMissedStatus, 96
com::toc::coredx::DDS::OfferedIncompatibleQosStatus, 97
com::toc::coredx::DDS::PublicationMatchedStatus, 104
com::toc::coredx::DDS::RequestedDeadlineMissedStatus, 121
com::toc::coredx::DDS::RequestedIncompatibleQosStatus, 122
com::toc::coredx::DDS::SampleLostStatus, 127
com::toc::coredx::DDS::SampleRejectedStatus, 128
com::toc::coredx::DDS::SubscriptionMatchedStatus, 142
get_trigger_value
com::toc::coredx::DDS::Condition, 16
get_type_name
com::toc::coredx::DDS::FooTypeSupport, 81
com::toc::coredx::DDS::MultiTopic, 94
com::toc::coredx::DDS::TopicDescription, 150
com::toc::coredx::DDS::TypeSupport, 158
get_view_state_mask
com::toc::coredx::DDS::ReadCondition, 118
getCTypeSupport
com::toc::coredx::DDS::TypeSupport, 158
group_data
com::toc::coredx::DDS::PublisherQos, 113
com::toc::coredx::DDS::SubscriberQos, 139
GuardCondition
com::toc::coredx::DDS::GuardCondition, 84
history,
com::toc::coredx::DDS::DataReaderQos, 28
com::toc::coredx::DDS::DataWriterQos, 37
com::toc::coredx::DDS::TopicQos, 154
ignore_participant
com::toc::coredx::DDS::DomainParticipant, 51
ignore_publication
com::toc::coredx::DDS::DomainParticipant, 51

ignore_subscription
com::toc::coredx::DDS::DomainParticipant, 51
ignore_topic
com::toc::coredx::DDS::DomainParticipant, 52
instance_handle
com::toc::coredx::DDS::SampleInfo, 125
instance_state
com::toc::coredx::DDS::SampleInfo, 125

latency_budget
com::toc::coredx::DDS::DataReaderQos, 28
com::toc::coredx::DDS::DataWriterQos, 37
com::toc::coredx::DDS::TopicQos, 154
lifespan
com::toc::coredx::DDS::DataWriterQos, 37
com::toc::coredx::DDS::TopicQos, 154
liveliness
com::toc::coredx::DDS::DataReaderQos, 28
com::toc::coredx::DDS::DataWriterQos, 37
com::toc::coredx::DDS::TopicQos, 154
lookup_datareader
com::toc::coredx::DDS::Subscriber, 134
lookup_datawriter
com::toc::coredx::DDS::Publisher, 108
lookup_instance
com::toc::coredx::DDS::FooDataReader, 71
com::toc::coredx::DDS::FooDataWriter, 79
lookup_participant
com::toc::coredx::DDS::DomainParticipantFactory, 55
lookup_topicdescription
com::toc::coredx::DDS::DomainParticipant, 52

no_writers_generation_count
com::toc::coredx::DDS::SampleInfo, 125

on_data_available
com::toc::coredx::DDS::DataReaderListener, 25
com::toc::coredx::DDS::DomainParticipantListener, 58
com::toc::coredx::DDS::SubscriberListener, 136
on_data_on_readers
com::toc::coredx::DDS::DomainParticipantListener, 58
com::toc::coredx::DDS::SubscriberListener, 136
on_inconsistent_topic
com::toc::coredx::DDS::DomainParticipantListener, 59
com::toc::coredx::DDS::TopicListener, 152
on_liveliness_changed
com::toc::coredx::DDS::DataReaderListener, 25
com::toc::coredx::DDS::DomainParticipantListener, 59
com::toc::coredx::DDS::SubscriberListener, 137
on_liveliness_lost
com::toc::coredx::DDS::DataWriterListener, 34
com::toc::coredx::DDS::DomainParticipantListener, 59
com::toc::coredx::DDS::PublisherListener, 111
on_offered_deadline_missed
com::toc::coredx::DDS::DataWriterListener, 34
com::toc::coredx::DDS::DomainParticipantListener, 59
com::toc::coredx::DDS::PublisherListener, 111
on_offered_incompatible_qos
com::toc::coredx::DDS::DataWriterListener, 35
com::toc::coredx::DDS::DomainParticipantListener, 59
com::toc::coredx::DDS::PublisherListener, 111
on_publication_matched
com::toc::coredx::DDS::DataWriterListener, 35
com::toc::coredx::DDS::DomainParticipantListener, 60
com::toc::coredx::DDS::PublisherListener, 112
on_requested_deadline_missed
com::toc::coredx::DDS::DataReaderListener, 26
com::toc::coredx::DDS::DomainParticipantListener, 60
com::toc::coredx::DDS::SubscriberListener, 137
on_requested_incompatible_qos
com::toc::coredx::DDS::DataReaderListener, 26
com::toc::coredx::DDS::DomainParticipantListener, 60

com::toc::coredx::DDS::SubscriberListener, 137
on_sample_lost
com::toc::coredx::DDS::DataReaderListener, 26
com::toc::coredx::DDS::DomainParticipantListener, 60
com::toc::coredx::DDS::SubscriberListener, 137
on_sample_rejected
com::toc::coredx::DDS::DataReaderListener, 26
com::toc::coredx::DDS::DomainParticipantListener, 61
com::toc::coredx::DDS::SubscriberListener, 137
on_subscription_matched
com::toc::coredx::DDS::DataReaderListener, 26
com::toc::coredx::DDS::DomainParticipantListener, 61
com::toc::coredx::DDS::SubscriberListener, 138
ownership
com::toc::coredx::DDS::DataReaderQos, 28
com::toc::coredx::DDS::DataWriterQos, 37
com::toc::coredx::DDS::TopicQos, 154
ownership_strength
com::toc::coredx::DDS::DataWriterQos, 37

partition
com::toc::coredx::DDS::PublisherQos, 113
com::toc::coredx::DDS::SubscriberQos, 139
peer_participants
com::toc::coredx::DDS::DomainParticipantQos, 62
presentation
com::toc::coredx::DDS::PublisherQos, 113
com::toc::coredx::DDS::SubscriberQos, 139
publication_handle
com::toc::coredx::DDS::SampleInfo, 125

qos_policy_str
com::toc::coredx::DDS::DDS, 39

read
com::toc::coredx::DDS::FooDataReader, 71
read_instance
com::toc::coredx::DDS::FooDataReader, 72
read_next_instance
com::toc::coredx::DDS::FooDataReader, 72
read_next_instance_w_condition
com::toc::coredx::DDS::FooDataReader, 73
read_next_sample
com::toc::coredx::DDS::FooDataReader, 73
read_w_condition
com::toc::coredx::DDS::FooDataReader, 73
reader_data_lifecycle
com::toc::coredx::DDS::DataReaderQos, 28
register_instance
com::toc::coredx::DDS::FooDataWriter, 79
register_instance_w_timestamp
com::toc::coredx::DDS::FooDataWriter, 79
register_type
com::toc::coredx::DDS::DomainParticipant, 52
com::toc::coredx::DDS::FooTypeSupport, 81
com::toc::coredx::DDS::TypeSupport, 158
reliability
com::toc::coredx::DDS::DataReaderQos, 28
com::toc::coredx::DDS::DataWriterQos, 37
com::toc::coredx::DDS::TopicQos, 154
resource_limits
com::toc::coredx::DDS::DataReaderQos, 28
com::toc::coredx::DDS::DataWriterQos, 37
com::toc::coredx::DDS::TopicQos, 154
resume_publications
com::toc::coredx::DDS::Publisher, 109
return_loan
com::toc::coredx::DDS::FooDataReader, 74

sample_rank
com::toc::coredx::DDS::SampleInfo, 125
sample_state
com::toc::coredx::DDS::SampleInfo, 125
set_default_datareader_qos
com::toc::coredx::DDS::Subscriber, 134
set_default_datawriter_qos
com::toc::coredx::DDS::Publisher, 109
set_default_participant_qos
com::toc::coredx::DDS::DomainParticipantFactory, 55
set_default_publisher_qos

com::toc::coredx::DDS::DomainParticipant, 52
set_default_subscriber_qos
com::toc::coredx::DDS::DomainParticipant, 52
set_default_topic_qos
com::toc::coredx::DDS::DomainParticipant, 53
set_enabled_statuses
com::toc::coredx::DDS::StatusCondition, 129
set_expression_parameters
com::toc::coredx::DDS::ContentFilteredTopic, 18
set_listener
com::toc::coredx::DDS::DataReader, 23
com::toc::coredx::DDS::DataWriter, 33
com::toc::coredx::DDS::DomainParticipant, 53
com::toc::coredx::DDS::Publisher, 109
com::toc::coredx::DDS::Subscriber, 135
com::toc::coredx::DDS::Topic, 147
set_qos
com::toc::coredx::DDS::DataReader, 23
com::toc::coredx::DDS::DataWriter, 33
com::toc::coredx::DDS::DomainParticipant, 53
com::toc::coredx::DDS::DomainParticipantFactory, 56
com::toc::coredx::DDS::Publisher, 109
com::toc::coredx::DDS::Subscriber, 135
com::toc::coredx::DDS::Topic, 147
set_query_parameters
com::toc::coredx::DDS::QueryCondition, 116
source_timestamp
com::toc::coredx::DDS::SampleInfo, 126
suspend_publications
com::toc::coredx::DDS::Publisher, 109

take
com::toc::coredx::DDS::FooDataReader, 74
take_instance
com::toc::coredx::DDS::FooDataReader, 75
take_next_instance
com::toc::coredx::DDS::FooDataReader, 76
take_next_instance_w_condition
com::toc::coredx::DDS::FooDataReader, 76
take_next_sample
com::toc::coredx::DDS::FooDataReader, 76
take_w_condition
com::toc::coredx::DDS::FooDataReader, 76
time_based_filter
com::toc::coredx::DDS::DataReaderQos, 28
topic_data
com::toc::coredx::DDS::TopicQos, 154
transport_priority
com::toc::coredx::DDS::DataWriterQos, 38
com::toc::coredx::DDS::TopicQos, 154

unregister_instance
com::toc::coredx::DDS::FooDataWriter, 79
unregister_instance_w_timestamp
com::toc::coredx::DDS::FooDataWriter, 80
user_data
com::toc::coredx::DDS::DataReaderQos, 28
com::toc::coredx::DDS::DataWriterQos, 38
com::toc::coredx::DDS::DomainParticipantQos, 62

valid_data
com::toc::coredx::DDS::SampleInfo, 126
view_state
com::toc::coredx::DDS::SampleInfo, 126

wait
com::toc::coredx::DDS::WaitSet, 161
wait_for_acknowledgments
com::toc::coredx::DDS::DataWriter, 33
com::toc::coredx::DDS::Publisher, 109
wait_for_historical_data
com::toc::coredx::DDS::DataReader, 24
WaitSet
com::toc::coredx::DDS::WaitSet, 160
write
com::toc::coredx::DDS::FooDataWriter, 80
write_w_timestamp
com::toc::coredx::DDS::FooDataWriter, 80
writer_data_lifecycle
com::toc::coredx::DDS::DataWriterQos, 38