CoreDX ™ Data Distribution Service

The leading Small Footprint DDS Middleware

C 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

Chapter 1

Overview

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 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 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 "C" language binding.

Intended Audience

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

cuses on the C programming language.

The reference documentation is organized in the following categories:

1. DDS Entities

   This includes the primary objects with which an application must interact to enable DDS publish-subscribe communications.

2. DDS Quality of Service

   This section documents the Quality of Service structures that configure the behavior of the CoreDX middleware.

3. DDS Conditions, Listeners, and WaitSets

   This section covers the various structures and concepts involved in delivering events to the application.

4. DDS Status Structures

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

5. CoreDX DDS DynamicTypes

   This includes routines and objects to support Dynamic Data Types.

Also, a list of API items which are not yet supported by CoreDX are provided here.

Chapter 2

Data Structure Documentation

2.1 DDS Entities

Classes

- struct DDS_DomainParticipantFactory

  The DDS_DomainParticipantFactory is used to configure, create and destroy DomainParticipant objects.

- struct DDS_DomainParticipant

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

- struct DDS_Subscriber

  The DDS_Subscriber configures, creates, manages and destroys DDS_DataReaders.

- struct DDS_Publisher

  The DDS_Publisher configures, creates, manages and destroys DDS_DataWriters.

- struct DDS_TopicDescription

  DDS_TopicDescription is an abstract 'class' that provides the foundation for DDS_Topic, DDS_-ContentFilteredTopic, and DDS_MultiTopic.

- struct DDS_Topic

  DDS_Topic is the basic description of data to be published or subscribed.

- struct DDS_ContentFilteredTopic

  DDS_ContentFilteredTopic provides a topic that may exclude data based on a specified filter. The Content-FilteredTopic is associated with another un-filtered topic ***related_topic****. It applies a filter to the data of the related topic. If a data sample passes the filter, it will be made available to a DataReader associated with the ContentFilteredTopic.*

- struct DDS_MultiTopic

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

- struct DDS_DataWriter

  The DDS_DataWriter entity provides an interface for the application to publish (write) data.

- struct DDS_DataReader

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

2.2 DDS Quality of Service

Classes

- struct CoreDX_RTPSWriterQosPolicy

  QoS Policy for configuring aspects of the RTPS Writer Protocol.

- struct CoreDX_RTPSReaderQosPolicy

  QoS Policy for configuring aspects of the RTPS Reader Protocol.

- struct CoreDX_DiscoveryQosPolicy

  QoS Policy for configuring aspects of the Discovery and Builtin entities.

- struct DDS_DomainParticipantFactoryQos

  Structure that holds DDS_DomainParticipantFactory Quality of Service policies.

- struct DDS_DomainParticipantQos

  Structure that holds DDS_DomainParticipant Quality of Service policies.

- struct DDS_TopicQos

  Structure that holds DDS_Topic Quality of Service policies.

- struct DDS_DataWriterQos

  Structure that holds DDS_DataWriter Quality of Service policies.

- struct DDS_PublisherQos

  Structure that holds DDS_Publisher Quality of Service policies.

- struct DDS_DataReaderQos

  Structure that holds DDS_DataReader Quality of Service policies.

- struct DDS_SubscriberQos

  Structure that holds DDS_Subscriber Quality of Service policies.

2.3 DDS Conditions, Listeners, and WaitSets

Modules

- DDS Listeners
- DDS Conditions
- DDS WaitSets

2.4 DDS Listeners

Classes

- struct DDS_TopicListener

  The DDS_TopicListener provides asynchronous notification of DDS_Topic events.

- struct DDS_TopicListener_cd

  The DDS_TopicListener_cd provides asynchronous notification of DDS_Topic events with additional callback data.

- struct DDS_DataWriterListener

  The DDS_DataWriterListener provides asynchronous notification of DDS_DataWriter events.

- struct DDS_DataWriterListener_cd

  The DDS_DataWriterListener_cd provides asynchronous notification of DDS_DataWriter events with additional callback data.

- struct DDS_PublisherListener

  The DDS_PublisherListener provides asynchronous notification of DDS_Publisher events.

- struct DDS_PublisherListener_cd

  The DDS_PublisherListener_cd provides asynchronous notification of DDS_Publisher events with additionall callback data.

- struct DDS_DataReaderListener

  The DDS_DataReaderListener provides asynchronous notification of DDS_DataReader events.

- struct DDS_DataReaderListener_cd

  The DDS_DataReaderListener_cd provides asynchronous notification of DDS_DataReader events with additional callback data.

- struct DDS_SubscriberListener

  The DDS_SubscriberListener provides asynchronous notification of DDS_Subscriber events.

- struct DDS_SubscriberListener_cd

  The DDS_SubscriberListener_cd provides asynchronous notification of DDS_Subscriber events with additional callback data.

- struct DDS_DomainParticipantListener

  The DDS_DomainParticipantListener provides asynchronous notification of DDS_DomainParticipant events.

- struct DDS_DomainParticipantListener_cd

The DDS_DomainParticipantListener_cd provides asynchronous notification of DDS_DomainParticipant events with additional callback data arguments.

2.5 DDS Conditions

Classes

- struct DDS_Condition

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

- struct DDS_GuardCondition

  *A DDS_GuardCondition is a condition where the **trigger_value** is under application control.*

- struct DDS_StatusCondition

  A DDS_StatusCondition is a condition associated with an Entity.

- struct DDS_ReadCondition

  A DDS_ReadCondition is a specialized DDS_Condition associated with a DDS_DataReader.

- struct DDS_QueryCondition

  A DDS_QueryCondition is a specialized DDS_ReadCondition which includes a filter.

2.6 DDS WaitSets

Classes

- struct DDS_WaitSet

  *A DDS_WaitSet maintains a set of DDS_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

- struct DDS_InconsistentTopicStatus

  Status related to the on_inconsistent_topic listener methods of the DDS_TopicListener structure.

- struct DDS_SampleLostStatus

  Status related to the on_sample_lost listener methods of the DDS_DataReader, DDS_Subscriber, and DDS_DomainParticipant structures.

- struct DDS_SampleRejectedStatus

  Status related to the on_sample_rejected listener methods of the DDS_DataReader, DDS_Subscriber, and DDS_DomainParticipant structures.

- struct DDS_LivelinessLostStatus

  Status related to the on_liveliness_lost listener methods of the DDS_DataWriter, DDS_Publisher, and DDS_DomainParticipant structures.

- struct DDS_LivelinessChangedStatus

  Status related to the on_liveliness_changed listener methods of the DDS_DataReader, DDS_Subscriber, and DDS_DomainParticipant structures.

- struct DDS_OfferedDeadlineMissedStatus

  Status related to the on_offered_deadline_missed listener methods of the DDS_DataWriter, DDS_Publisher, and DDS_DomainParticipant structures.

- struct DDS_RequestedDeadlineMissedStatus

  Status related to the on_requested_deadline_missed listener methods of the DDS_DataReader, DDS_Subscriber, and DDS_DomainParticipant structures.

- struct DDS_OfferedIncompatibleQosStatus

  Status related to the on_offered_incompatible_qos listener methods of the DDS_DataWriter, DDS_Publisher, and DDS_DomainParticipant structures.

- struct DDS_RequestedIncompatibleQosStatus

  Status related to the on_requested_incompatible_qos listener methods of the DDS_DataReader, DDS_Subscriber, and DDS_DomainParticipant structures.

- struct DDS_PublicationMatchedStatus

  Status related to the on_publication_matched listener methods of the DDS_DataWriter, DDS_Publisher, and DDS_DomainParticipant structures.

- struct DDS_SubscriptionMatchedStatus

Status related to the on_subscription_matched listener methods of the DDS_DataReader, DDS_Subscriber, and DDS_DomainParticipant structures.

2.7.1 Detailed Description

- DDS_InconsistentTopicStatus
- DDS_LivelinessChangedStatus
- DDS_LivelinessLostStatus
- DDS_OfferedDeadlineMissedStatus
- DDS_OfferedIncompatibleQosStatus
- DDS_PublicationMatchedStatus
- DDS_RequestedDeadlineMissedStatus
- DDS_RequestedIncompatibleQosStatus
- DDS_SampleLostStatus
- DDS_SampleRejectedStatus
- DDS_SubscriptionMatchedStatus

Chapter 3

API Documentation

3.1 DDS_Condition Struct Reference

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

Related Functions

(Note that these are not member functions.)

- unsigned char DDS_Condition_get_trigger_value (DDS_Condition c)
  *This routine returns the current value of the **trigger_value** in Condition **c**.*

3.1.1 Detailed Description

A DDS_Condition can be added to a DDS_WaitSet to provide synchronous event notification. A DDS_-Condition has a **trigger_value** which can be TRUE or FALSE.

3.1.2 Friends And Related Function Documentation

3.1.2.1 unsigned char DDS_Condition_get_trigger_value (DDS_Condition *c*) [related]

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 struct was generated from the following file:

- dds.h

3.2 DDS_ContentFilteredTopic Struct Reference

DDS_ContentFilteredTopic provides a topic that may exclude data based on a specified filter. The ContentFilteredTopic is associated with another un-filtered topic **related_topic**. It applies a filter to the data of the related topic. If a data sample passes the filter, it will be made available to a DataReader associated with the ContentFilteredTopic.

Related Functions

(Note that these are not member functions.)

- DDS_TopicDescription DDS_ContentFilteredTopic_TopicDescription (DDS_ContentFilteredTopic t)

  This operation 'casts' the provide DDS_ContentFilteredTopic to a DDS_TopicDescrition.

- DDS_Topic DDS_ContentFilteredTopic_get_related_topic (DDS_ContentFilteredTopic t)

  This returns the real Topic associated with the ContentFilteredTopic.

- DDS_ReturnCode_t DDS_ContentFilteredTopic_get_expression_parameters (DDS_ContentFilteredTopic t, DDS_StringSeq *parameters)

  This accesses the current set of parameters used by the ContentFilteredTopic.

- DDS_ReturnCode_t DDS_ContentFilteredTopic_set_expression_parameters (DDS_ContentFilteredTopic t, const DDS_StringSeq *parameters)

  This specifies a new set of parameters for use with the filter_expression.

3.2.1 Detailed Description

DDS_ContentFilteredTopic provides a topic that may exclude data based on a specified filter. The ContentFilteredTopic is associated with another un-filtered topic **related_topic**. It applies a filter to the data of the related topic. If a data sample passes the filter, it will be made available to a DataReader associated with the ContentFilteredTopic. The **filter_expression** is an SQL like condition expression, and **filter_parameters** provide optional parameters that are referenced by the **filter_expression**. The syntax of the filter expression is similar to the WHERE clause in SQL. For example "x<4" is a valid filter expression. It would test that data member 'x' is less than value '4'. If the filter expression evaluates to TRUE, then the data sample will be available, otherwise the data sample would be 'filtered' (excluded).

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

CoreDX DDS also includes support for the '**IN**' operator. This provides a very powerful mechanism for testing that a value appears in a set of values. For example "symbol IN ('ge', 'msft', 'ibm')" will select all samples that have a symbol value of 'ge', 'msft', or 'ibm'. This could also be written as a series of equality

tests combined with the OR operator; however, the IN operator is much more efficient. A filter that matches on several hundred or even thousands of values can be implemented very efficiently using the 'IN' operator.

The filter_expression can refer to parameters. The syntax for 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.

See also

DDS_DomainParticipant_create_contentfilteredtopic()

3.2.2 Friends And Related Function Documentation

3.2.2.1 DDS_ReturnCode_t DDS_ContentFilteredTopic_get_expression_parameters (DDS_ContentFilteredTopic *t*, DDS_StringSeq * *parameters*) [related]

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 DDS_Topic DDS_ContentFilteredTopic_get_related_topic (DDS_ContentFilteredTopic *t*) [related]

This returns the real Topic associated with the ContentFilteredTopic.

That is, the Topic provided when the ContentFilteredTopic was created.

3.2.2.3 DDS_ReturnCode_t DDS_ContentFilteredTopic_set_expression_parameters (DDS_ContentFilteredTopic *t*, const DDS_StringSeq * *parameters*) [related]

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 struct was generated from the following file:

- dds.h

3.3 CoreDX_DiscoveryQosPolicy Struct Reference

QoS Policy for configuring aspects of the Discovery and Builtin entities.

Public Attributes

- DDS_DiscoveryQosPolicyDiscoveryKind discovery_kind

  Type of disocvery to use for this Entity.

- DDS_BUILTIN_TOPIC_KEY_TYPE_NATIVE guid_pid

  Override the default value of 'pid' in Discovery GUID. ['0' means use default].

- DDS_Duration_t dpd_push_period
- DDS_Duration_t dpd_lease_duration
- unsigned char send_initial_nack

3.3.1 Detailed Description

QoS Policy for configuring aspects of the Discovery and Builtin entities.

3.3.2 Member Data Documentation

3.3.2.1 DDS_Duration_t CoreDX_DiscoveryQosPolicy::dpd_lease_duration

How long we consider a discoverd Participant to be alive without hearing from them

3.3.2.2 DDS_Duration_t CoreDX_DiscoveryQosPolicy::dpd_push_period

Multicast DiscoveredParticipantData each period

3.3.2.3 unsigned char CoreDX_DiscoveryQosPolicy::send_initial_nack

send a ACKNACK immediately after matching with new Writer.

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

- dds.h

3.4 CoreDX_RTPSReaderQosPolicy Struct Reference

QoS Policy for configuring aspects of the RTPS Reader Protocol.

Public Attributes

- unsigned char accept_batch_msg
- unsigned char send_typecode
- unsigned char send_initial_nack

3.4.1 Detailed Description

QoS Policy for configuring aspects of the RTPS Reader Protocol.

3.4.2 Member Data Documentation

3.4.2.1 unsigned char CoreDX_RTPSReaderQosPolicy::accept_batch_msg

allow writers to use the 'BATCH' RTPS message

3.4.2.2 unsigned char CoreDX_RTPSReaderQosPolicy::send_initial_nack

send a ACKNACK immediately after matching with new Writer.

3.4.2.3 unsigned char CoreDX_RTPSReaderQosPolicy::send_typecode

send 'typecode' information for associated data type.

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

- dds.h

3.5 CoreDX_RTPSWriterQosPolicy Struct Reference

QoS Policy for configuring aspects of the RTPS Writer Protocol.

Public Attributes

- DDS_Duration_t ack_deadline
- unsigned int min_buffer_size
- unsigned int max_buffer_size
- unsigned char apply_filters
- unsigned char enable_batch_msg
- unsigned char send_typecode

3.5.1 Detailed Description

QoS Policy for configuring aspects of the RTPS Writer Protocol.

3.5.2 Member Data Documentation

3.5.2.1 DDS_Duration_t CoreDX_RTPSWriterQosPolicy::ack_deadline

after which a realiable reader will be considered unresponsive

3.5.2.2 unsigned char CoreDX_RTPSWriterQosPolicy::apply_filters

apply ContentFilter(s) at the writer (writer side filtering)

3.5.2.3 unsigned char CoreDX_RTPSWriterQosPolicy::enable_batch_msg

use the 'BATCH' RTPS message to send data if all Readers accept BATCH

3.5.2.4 unsigned int CoreDX_RTPSWriterQosPolicy::max_buffer_size

max size in bytes of written data

3.5.2.5 unsigned int CoreDX_RTPSWriterQosPolicy::min_buffer_size

min size in bytes of written data

3.5.2.6 unsigned char CoreDX_RTPSWriterQosPolicy::send_typecode

send 'typecode' information for associated data type.

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

- dds.h

3.6 DDS_DataReader Struct Reference

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

Related Functions

(Note that these are not member functions.)

- DDS_ReturnCode_t DDS_DataReader_enable (DDS_DataReader dr)

  Enables the DDS_DataReader.

- DDS_InstanceHandle_t DDS_DataReader_get_instance_handle (DDS_DataReader dr)

  This operation returns the InstanceHandle_t that identifies the DataReader.

- DDS_StatusMask DDS_DataReader_get_status_changes (DDS_DataReader dr)

  *This returns the list of **triggered** communication statuses in the DataReader.*

- DDS_ReturnCode_t DDS_DataReader_delete_contained_entities (DDS_DataReader dr)

  This operation deletes all the ReadCondition and QueryCondition objects previously created by means of the DDS_DataReader_create_readcondition() and DDS_DataReader_create_querycondition() operations.

- DDS_ReturnCode_t DDS_DataReader_set_qos (DDS_DataReader dr, const DDS_DataReaderQos *qos)

  Sets the DDS_DataReaderQos values.

- DDS_ReturnCode_t DDS_DataReader_get_qos (DDS_DataReader dr, DDS_DataReaderQos *qos)

  *Returns the current DDS_DataReaderQos settings held in the DataReader **dr**.*

- DDS_ReturnCode_t DDS_DataReader_set_listener (DDS_DataReader dr, DDS_DataReaderListener *a_listener, DDS_StatusMask mask)

  *Installs a DDS_DataReaderListener on DataReader **dr**.*

- DDS_ReturnCode_t DDS_DataReader_set_listener_cd (DDS_DataReader dr, DDS_DataReaderListener_cd *a_listener, DDS_StatusMask mask, void *callback_data)

  *Installs a DDS_DataReaderListener_cd on DataReader **dr**.*

- DDS_DataReaderListener * DDS_DataReader_get_listener (DDS_DataReader dr)

  This operation returns the currently installed DDS_DataReaderListener.

- DDS_DataReaderListener_cd * DDS_DataReader_get_listener_cd (DDS_DataReader dr)

  This operation returns the currently installed DDS_DataReaderListener_cd.

- DDS_TopicDescription DDS_DataReader_get_topicdescription (DDS_DataReader dr)

  *Returns the DDS_TopicDescription associated with DataReader **dr**.*

- DDS_Subscriber DDS_DataReader_get_subscriber (DDS_DataReader dr)

  *Returns the DDS_Subscribier that contains DataReader **dr**.*

- DDS_ReturnCode_t DDS_DataReader_wait_for_historical_data (DDS_DataReader dr, const DDS_Duration_t *max_wait)

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

- DDS_StatusCondition DDS_DataReader_get_statuscondition (DDS_DataReader dr)

  This operation allows access to the DDS_StatusCondition associated with the DataReader.

- DDS_ReadCondition DDS_DataReader_create_readcondition (DDS_DataReader dr, DDS_SampleStateMask sample_states, DDS_ViewStateMask view_states, DDS_InstanceStateMask instance_states)

  Creates a DDS_ReadCondition that is associated with this DataReader.

- DDS_ReturnCode_t DDS_DataReader_delete_readcondition (DDS_DataReader dr, DDS_ReadCondition a_condition)

  Destroys a DDS_ReadCondition (or DDS_QueryCondition).

- DDS_QueryCondition DDS_DataReader_create_querycondition (DDS_DataReader dr, DDS_SampleStateMask sample_states, DDS_ViewStateMask view_states, DDS_InstanceStateMask instance_states, const char *query_expression, const DDS_StringSeq *query_parameters)

  Creates a DDS_QueryCondition.

- DDS_ReturnCode_t DDS_DataReader_get_matched_publications (DDS_DataReader dr, DDS_InstanceHandleSeq *publication_handles)

  *This operation retrieves the list of DataWriters currently matched with the DataReader **dr**.*

- DDS_ReturnCode_t DDS_DataReader_get_matched_publication_data (DDS_DataReader dr, DDS_PublicationBuiltinTopicData *publication_data, const DDS_InstanceHandle_t publication_handle)

  *This operation returns data that describes a particular matched DataWriter identified by **publication_handle**.*

- DDS_ReturnCode_t DDS_DataReader_return_loan (DDS_DataReader dr, DDS_PointerSeq *received_data, DDS_SampleInfoSeq *sample_infos)

  Returns data and sample_info values to a DataReader.

- DDS_ReturnCode_t DDS_DataReader_get_sample_rejected_status (DDS_DataReader dr, DDS_SampleRejectedStatus *status)

  Provides access to the current DDS_SampleRejectedStatus of the DataReader.

- DDS_ReturnCode_t DDS_DataReader_get_liveliness_changed_status (DDS_DataReader dr, DDS_LivelinessChangedStatus *status)

  Provides access to the current DDS_LivelinessChangedStatus of the DataReader.

- DDS_ReturnCode_t DDS_DataReader_get_requested_deadline_missed_status (DDS_DataReader dr, DDS_RequestedDeadlineMissedStatus *status)

  Provides access to the current DDS_RequestedDeadlineMissedStatus of the DataReader.

- DDS_ReturnCode_t DDS_DataReader_get_requested_incompatible_qos_status (DDS_DataReader dr, DDS_RequestedIncompatibleQosStatus *status)

  Provides access to the current DDS_RequestedIncompatibleQosStatus of the DataReader.

- DDS_ReturnCode_t DDS_DataReader_get_subscription_matched_status (DDS_DataReader dr, DDS_SubscriptionMatchedStatus *status)

  Provides access to the current DDS_SubscriptionMatchedStatus of the DataReader.

- DDS_ReturnCode_t DDS_DataReader_get_sample_lost_status (DDS_DataReader dr, DDS_SampleLostStatus *status)

  Provides access to the current DDS_SampleLostStatus of the DataReader.

- DDS_ReturnCode_t DDS_DataReader_get_key_value (DDS_DataReader dr, void *key_holder, DDS_InstanceHandle_t handle)

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

- DDS_InstanceHandle_t DDS_DataReader_lookup_instance (DDS_DataReader dr, void *instance_data)

  *Returns the handle that identifies the data instance provided in **instance_data**.*

- DDS_ReturnCode_t DDS_DataReader_read (DDS_DataReader dr, DDS_PointerSeq *received_data, DDS_SampleInfoSeq *sample_infos, int max_samples, DDS_SampleStateMask sample_states, DDS_ViewStateMask view_states, DDS_InstanceStateMask instance_states)

  This operation accesses a collection of data values (with associated DDS_SampleInfo) within the DataReader.

- DDS_ReturnCode_t DDS_DataReader_take (DDS_DataReader dr, DDS_PointerSeq *received_data, DDS_SampleInfoSeq *sample_infos, int max_samples, DDS_SampleStateMask sample_states, DDS_ViewStateMask view_states, DDS_InstanceStateMask instance_states)

  This operation takes a collection of data values (with associated DDS_SampleInfo) from the DataReader.

- DDS_ReturnCode_t DDS_DataReader_read_w_condition (DDS_DataReader dr, DDS_PointerSeq *received_data, DDS_SampleInfoSeq *sample_infos, int max_samples, DDS_ReadCondition a_condition)

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

- DDS_ReturnCode_t DDS_DataReader_take_w_condition (DDS_DataReader dr, DDS_PointerSeq *received_data, DDS_SampleInfoSeq *sample_infos, int max_samples, DDS_ReadCondition a_condition)

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

- DDS_ReturnCode_t DDS_DataReader_read_next_sample (DDS_DataReader dr, void *received_data, DDS_SampleInfo *sample_info)

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

- DDS_ReturnCode_t DDS_DataReader_take_next_sample (DDS_DataReader dr, void *received_data, DDS_SampleInfo *sample_info)

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

- DDS_ReturnCode_t DDS_DataReader_read_instance (DDS_DataReader dr, DDS_PointerSeq *received_data, DDS_SampleInfoSeq *sample_infos, int max_samples, DDS_InstanceHandle_t a_handle, DDS_SampleStateMask sample_states, DDS_ViewStateMask view_states, DDS_InstanceStateMask instance_states)

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

- DDS_ReturnCode_t DDS_DataReader_take_instance (DDS_DataReader dr, DDS_PointerSeq *received_data, DDS_SampleInfoSeq *sample_infos, int max_samples, DDS_InstanceHandle_t a_handle, DDS_SampleStateMask sample_states, DDS_ViewStateMask view_states, DDS_InstanceStateMask instance_states)

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

- DDS_ReturnCode_t DDS_DataReader_read_next_instance (DDS_DataReader dr, DDS_PointerSeq *received_data, DDS_SampleInfoSeq *sample_infos, int max_samples, DDS_InstanceHandle_t previous_handle, DDS_SampleStateMask sample_states, DDS_ViewStateMask view_states, DDS_InstanceStateMask instance_states)

  This operation accesses a collection of data values (with associated DDS_SampleInfo), instance by instance, within the DataReader.

- DDS_ReturnCode_t DDS_DataReader_take_next_instance (DDS_DataReader dr, DDS_PointerSeq *received_data, DDS_SampleInfoSeq *sample_infos, int max_samples, DDS_InstanceHandle_t previous_handle, DDS_SampleStateMask sample_states, DDS_ViewStateMask view_states, DDS_InstanceStateMask instance_states)

  This operation takes a collection of data values (with associated DDS_SampleInfo), instance by instance, from the DataReader.

- DDS_ReturnCode_t DDS_DataReader_read_next_instance_w_condition (DDS_DataReader dr, DDS_PointerSeq *received_data, DDS_SampleInfoSeq *sample_infos, int max_samples, DDS_InstanceHandle_t previous_handle, DDS_ReadCondition a_condition)

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

- DDS_ReturnCode_t DDS_DataReader_take_next_instance_w_condition (DDS_DataReader dr, DDS_PointerSeq *received_data, DDS_SampleInfoSeq *sample_infos, int max_samples, DDS_InstanceHandle_t previous_handle, DDS_ReadCondition a_condition)

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

3.6.1 Detailed Description

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

3.6.2 Friends And Related Function Documentation

3.6.2.1 DDS_QueryCondition DDS_DataReader_create_querycondition (DDS_DataReader *dr*, DDS_SampleStateMask *sample_states*, DDS_ViewStateMask *view_states*, DDS_InstanceStateMask *instance_states*, const char * *query_expression*, const DDS_StringSeq * *query_parameters*) [related]

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

DDS_QueryCondition
DDS_DataReader_read_w_condition()
DDS_DataReader_take_w_condition()

3.6.2.2 DDS_ReadCondition DDS_DataReader_create_readcondition (DDS_DataReader *dr,* DDS_SampleStateMask *sample_states,* DDS_ViewStateMask *view_states,* DDS_InstanceStateMask *instance_states*) [related]

Creates a DDS_ReadCondition that is associated with this DataReader.

The returned condition can be added to a DDS_WaitSet or used in a call to the specialized **read()** or **take()** operations. For example see DDS_DataReader_read_w_condition().

3.6.2.3 DDS_ReturnCode_t DDS_DataReader_delete_contained_entities (DDS_DataReader *dr*) [related]

This operation deletes all the ReadCondition and QueryCondition objects previously created by means of the DDS_DataReader_create_readcondition() and DDS_DataReader_create_querycondition() operations.

After successful execution, the application may delete the Publisher by calling DDS_Subscriber_delete_datareader().

If any of the objects cannot be deleted, this routine will return DDS_RETCODE_PRECONDITION_NOT_MET.

3.6.2.4 DDS_ReturnCode_t DDS_DataReader_delete_readcondition (DDS_DataReader *dr,* DDS_ReadCondition *a_condition*) [related]

Destroys a DDS_ReadCondition (or DDS_QueryCondition).

The provided **a_condition** must have been previously created via a call to DDS_DataReader_create_readcondition() or DDS_DataReader_create_querycondition().

The **a_condition** argument must belong to DataReader **dr**. Otherwise, the error DDS_RETCODE_PRECONDITION_NOT_MET will be returned.

If the DataReader is actively processing the ReadCondition, this routine will return DDS_RETCODE_ERROR; in this case, the delete_readcondition() call should be re-tried.

3.6.2.5 DDS_ReturnCode_t DDS_DataReader_enable (DDS_DataReader *dr*) [related]

Enables the DDS_DataReader.

A DataReader is created either enabled or not based on the DDS_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 DDS_NOT_ENABLED error. DDS_DataReader_enable() may be called on an already enabled DataReader [it will have no effect].

3.6.2.6 DDS_ReturnCode_t DDS_DataReader_get_key_value (DDS_DataReader *dr*, void * *key_holder*, DDS_InstanceHandle_t *handle*) [related]

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

Note

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.

3.6.2.7 DDS_DataReaderListener * DDS_DataReader_get_listener (DDS_DataReader *dr*) [related]

This operation returns the currently installed DDS_DataReaderListener.

Note

Because the infrastructure makes a copy of the listener provided in DDS_DataReader_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.

3.6.2.8 DDS_DataReaderListener_cd * DDS_DataReader_get_listener_cd (DDS_DataReader *dr*) [related]

This operation returns the currently installed DDS_DataReaderListener_cd.

Note

Because the infrastructure holds a pointer to the listener provided in DDS_DataReader_set_listener(), the returned structure pointer will match the pointer originally provided. The application should not free the data referenced by the returned pointer.

3.6.2.9 DDS_ReturnCode_t DDS_DataReader_get_liveliness_changed_status (DDS_DataReader *dr,* DDS_LivelinessChangedStatus * *status*) [related]

Provides access to the current DDS_LivelinessChangedStatus of the DataReader.

As a side-effect, this routine will reset the **total_count_change** status field to zero.

3.6.2.10 DDS_ReturnCode_t DDS_DataReader_get_matched_publication_data (DDS_DataReader *dr,* DDS_PublicationBuiltinTopicData * *publication_data,* const DDS_InstanceHandle_t *publication_handle*) [related]

This operation returns data that describes a particular matched DataWriter identified by **publication_handle**.

An appropriate handle can be obtained through a call to DDS_DataReader_get_matched_publications().

If **publication_handle** does not identify a matched DataWriter, this routine will return DDS_RETCODE_-PRECONDITION_NOT_MET.

3.6.2.11 DDS_ReturnCode_t DDS_DataReader_get_matched_publications (DDS_DataReader *dr,* DDS_InstanceHandleSeq * *publication_handles*) [related]

This operation retrieves the list of DataWriters currently matched with the 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 DDS_DomainParticipant_ignore_publication(), then it will not appear in the list.

3.6.2.12 DDS_ReturnCode_t DDS_DataReader_get_qos (DDS_DataReader *dr,* DDS_DataReaderQos * *qos*) [related]

Returns the current DDS_DataReaderQos settings held in the DataReader **dr**.

This routines copies data from the DataReader QoS properties into **qos**.

Note

The qos structure may contain sequences or strings that are populated with dynamic memory. The caller is responsible for freeing the dynamic memory of these items.

For example, the sequence 'qos->user_data' may have dynamically allocated memory assigned. This can be released by a call to **seq_clear(&qos->user_data)**.

3.6.2.13 DDS_ReturnCode_t DDS_DataReader_get_requested_deadline_missed_status (DDS_DataReader *dr,* DDS_RequestedDeadlineMissedStatus * *status*) [related]

Provides access to the current DDS_RequestedDeadlineMissedStatus of the DataReader.

As a side-effect, this routine will reset the **total_count_change** status field to zero.

3.6.2.14 DDS_ReturnCode_t DDS_DataReader_get_requested_incompatible_qos_status (DDS_DataReader *dr,* DDS_RequestedIncompatibleQosStatus * *status*) [related]

Provides access to the current DDS_RequestedIncompatibleQosStatus of the DataReader.

As a side-effect, this routine will reset the **total_count_change** status field to zero.

3.6.2.15 DDS_ReturnCode_t DDS_DataReader_get_sample_lost_status (DDS_DataReader *dr,* DDS_SampleLostStatus * *status*) [related]

Provides access to the current DDS_SampleLostStatus of the DataReader.

As a side-effect, this routine will reset the **total_count_change** status field to zero.

3.6.2.16 DDS_ReturnCode_t DDS_DataReader_get_sample_rejected_status (DDS_DataReader *dr,* DDS_SampleRejectedStatus * *status*) [related]

Provides access to the current DDS_SampleRejectedStatus of the DataReader.

As a side-effect, this routine will reset the **total_count_change** status field to zero.

3.6.2.17 DDS_StatusMask DDS_DataReader_get_status_changes (DDS_DataReader *dr*) [related]

This returns the list of **triggered** communication statuses in the DataReader.

If the DataReader is not enabled, all statuses will be **untriggered**.

3.6.2.18 DDS_StatusCondition DDS_DataReader_get_statuscondition (DDS_DataReader *dr*) [related]

This operation allows access to the DDS_StatusCondition associated with the DataReader.

The returned condition can be added to a DDS_WaitSet.

3.6.2.19 DDS_ReturnCode_t DDS_DataReader_get_subscription_matched_status (DDS_DataReader *dr,* DDS_SubscriptionMatchedStatus * *status*) [related]

Provides access to the current DDS_SubscriptionMatchedStatus of the DataReader.

As a side-effect, this routine will reset the **total_count_change** status field to zero.

3.6.2.20 DDS_TopicDescription DDS_DataReader_get_topicdescription (DDS_DataReader *dr*) [related]

Returns the DDS_TopicDescription associated with DataReader **dr**.

The returned TopicDescription is really a DDS_Topic, DDS_ContentFilteredTopic or DDS_MultiTopic.

3.6.2.21 DDS_InstanceHandle_t DDS_DataReader_lookup_instance (DDS_DataReader *dr,* void * *instance_data*) [related]

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.

Note

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.

3.6.2.22 DDS_ReturnCode_t DDS_DataReader_read (DDS_DataReader *dr,* DDS_PointerSeq * *received_data,* DDS_SampleInfoSeq * *sample_infos,* int *max_samples,* DDS_SampleStateMask *sample_states,* DDS_ViewStateMask *view_states,* DDS_InstanceStateMask *instance_states*) [related]

This operation accesses a collection of data values (with associated DDS_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 DDS_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 DDS_DataReader_return_loan().

The **read()** operation will set the DDS_SampleInfo::sample_state to DDS_READ_SAMPLE_STATE.

The **read()** operation may set the DDS_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 DDS_RETCODE_NO_DATA.

Note

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.

3.6.2.23 DDS_ReturnCode_t DDS_DataReader_read_instance (DDS_DataReader *dr*, DDS_PointerSeq * *received_data*, DDS_SampleInfoSeq * *sample_infos*, int *max_samples*, DDS_InstanceHandle_t *a_handle*, DDS_SampleStateMask *sample_states*, DDS_ViewStateMask *view_states*, DDS_InstanceStateMask *instance_states*) [related]

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

Note

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

DDS_DataReader_read()

3.6.2.24 DDS_ReturnCode_t DDS_DataReader_read_next_instance (DDS_DataReader *dr*, DDS_PointerSeq * *received_data*, DDS_SampleInfoSeq * *sample_infos*, int *max_samples*, DDS_InstanceHandle_t *previous_handle*, DDS_SampleStateMask *sample_states*, DDS_ViewStateMask *view_states*, DDS_InstanceStateMask *instance_states*) [related]

This operation accesses a collection of data values (with associated DDS_SampleInfo), instance by instance, within the DataReader.

Note

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

DDS_DataReader_read()

3.6.2.25 DDS_ReturnCode_t DDS_DataReader_read_next_instance_w_condition (DDS_DataReader *dr*, DDS_PointerSeq * *received_data*, DDS_SampleInfoSeq * *sample_infos*, int *max_samples*, DDS_InstanceHandle_t *previous_handle*, DDS_ReadCondition *a_condition*) [related]

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

Note

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

DDS_DataReader_read()

3.6.2.26 DDS_ReturnCode_t DDS_DataReader_read_next_sample (DDS_DataReader *dr*, void * *received_data*, DDS_SampleInfo * *sample_info*) [related]

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

Note

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

DDS_DataReader_read()

3.6.2.27 DDS_ReturnCode_t DDS_DataReader_read_w_condition (DDS_DataReader *dr*, DDS_PointerSeq * *received_data*, DDS_SampleInfoSeq * *sample_infos*, int *max_samples*, DDS_ReadCondition *a_condition*) [related]

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

Note

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

DDS_DataReader_read()

3.6.2.28 DDS_ReturnCode_t DDS_DataReader_return_loan (DDS_DataReader *dr*, DDS_PointerSeq * *received_data*, DDS_SampleInfoSeq * *sample_infos*) [related]

Returns data and sample_info values to a DataReader.

When an application calls DataReader **read()** or **take()** operations, these routines may 'loan' data and DDS_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 called only if previous **read()** or **take()** calls 'loaned' data to the application. See DDS_DataReader_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 DDS_RETCODE_PRECONDITION_NOT_MET will be returned.

Note

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

3.6.2.29 DDS_ReturnCode_t DDS_DataReader_set_listener (DDS_DataReader *dr*, DDS_DataReaderListener * *a_listener*, DDS_StatusMask *mask*) [related]

Installs a DDS_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.

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

3.6.2.30 DDS_ReturnCode_t DDS_DataReader_set_listener_cd (DDS_DataReader *dr*, DDS_DataReaderListener_cd * *a_listener*, DDS_StatusMask *mask*, void * *callback_data*) [related]

Installs a DDS_DataReaderListener_cd on DataReader **dr**.

Only one listener may be attached to a DataReader at a time. A call to **set_listener_cd()** will replace any current listener with **a_listener**.

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

The infrastructure will not hold a pointer to the listener structure which means that it must be persisted by the application.

3.6.2.31 DDS_ReturnCode_t DDS_DataReader_set_qos (DDS_DataReader *dr*, const DDS_DataReaderQos * *qos*) [related]

Sets the DDS_DataReaderQos values.

These QoS values affect the behavior of the DDS_DataReader.

3.6.2.32 DDS_ReturnCode_t DDS_DataReader_take (DDS_DataReader *dr*, DDS_PointerSeq * *received_data*, DDS_SampleInfoSeq * *sample_infos*, int *max_samples*, DDS_SampleStateMask *sample_states*, DDS_ViewStateMask *view_states*, DDS_InstanceStateMask *instance_states*) [related]

This operation takes a collection of data values (with associated DDS_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 DDS_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 DDS_DataReader_return_loan().

The **take()** operation will set the DDS_SampleInfo::sample_state to DDS_READ_SAMPLE_STATE.

The **take()** operation may set the DDS_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 DDS_RETCODE_NO_DATA.

Note

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.

3.6.2.33 DDS_ReturnCode_t DDS_DataReader_take_instance (DDS_DataReader *dr*, DDS_PointerSeq * *received_data*, DDS_SampleInfoSeq * *sample_infos*, int *max_samples*, DDS_InstanceHandle_t *a_handle*, DDS_SampleStateMask *sample_states*, DDS_ViewStateMask *view_states*, DDS_InstanceStateMask *instance_states*) [related]

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

Note

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

DDS_DataReader_take()

3.6.2.34 DDS_ReturnCode_t DDS_DataReader_take_next_instance (DDS_DataReader *dr*, DDS_PointerSeq * *received_data*, DDS_SampleInfoSeq * *sample_infos*, int *max_samples*, DDS_InstanceHandle_t *previous_handle*, DDS_SampleStateMask *sample_states*, DDS_ViewStateMask *view_states*, DDS_InstanceStateMask *instance_states*) `[related]`

This operation takes a collection of data values (with associated DDS_SampleInfo), instance by instance, from the DataReader.

Note

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

DDS_DataReader_take()

3.6.2.35 DDS_ReturnCode_t DDS_DataReader_take_next_instance_w_condition (DDS_DataReader *dr*, DDS_PointerSeq * *received_data*, DDS_SampleInfoSeq * *sample_infos*, int *max_samples*, DDS_InstanceHandle_t *previous_handle*, DDS_ReadCondition *a_condition*) `[related]`

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

Note

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

DDS_DataReader_take()

3.6.2.36 DDS_ReturnCode_t DDS_DataReader_take_next_sample (DDS_DataReader *dr*, void * *received_data*, DDS_SampleInfo * *sample_info*) [related]

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

Note

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

DDS_DataReader_take()

3.6.2.37 DDS_ReturnCode_t DDS_DataReader_take_w_condition (DDS_DataReader *dr*, DDS_PointerSeq * *received_data*, DDS_SampleInfoSeq * *sample_infos*, int *max_samples*, DDS_ReadCondition *a_condition*) [related]

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

Note

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

DDS_DataReader_take()

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

- dds.h

3.7 DDS_DataReaderListener Struct Reference

The DDS_DataReaderListener provides asynchronous notification of DDS_DataReader events.

Public Attributes

- void(∗ on_requested_deadline_missed)(DDS_DataReader the_reader, DDS_RequestedDeadlineMissedStatus status)
- void(∗ on_requested_incompatible_qos)(DDS_DataReader the_reader, DDS_RequestedIncompatibleQosStatus status)
- void(∗ on_sample_rejected)(DDS_DataReader the_reader, DDS_SampleRejectedStatus status)
- void(∗ on_liveliness_changed)(DDS_DataReader the_reader, DDS_LivelinessChangedStatus status)
- void(∗ on_data_available)(DDS_DataReader the_reader)
- void(∗ on_subscription_matched)(DDS_DataReader the_reader, DDS_SubscriptionMatchedStatus status)
- void(∗ on_sample_lost)(DDS_DataReader the_reader, DDS_SampleLostStatus status)

3.7.1 Detailed Description

The DDS_DataReaderListener provides asynchronous notification of DDS_DataReader events. This listener can be installed during DataReader creation, DDS_Subscriber_create_datareader(), as well as by calling DDS_DataReader_set_listener().

Note

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

3.7.2 Member Data Documentation

3.7.2.1 void(∗ DDS_DataReaderListener::on_data_available)(DDS_DataReader the_reader)

on_data_available() is 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.

3.7.2.2 void(∗ DDS_DataReaderListener::on_liveliness_changed)(DDS_DataReader the_reader, DDS_LivelinessChangedStatus status)

on_liveliness_changed() is 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.

3.7.2.3 void(* DDS_DataReaderListener::on_requested_deadline_missed)(DDS_DataReader the_reader, DDS_RequestedDeadlineMissedStatus status)

on_requested_deadline_missed() is 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.

3.7.2.4 void(* DDS_DataReaderListener::on_requested_incompatible_qos)(DDS_DataReader the_reader, DDS_RequestedIncompatibleQosStatus status)

on_requested_incompatible_qos() is 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.

3.7.2.5 void(* DDS_DataReaderListener::on_sample_lost)(DDS_DataReader the_reader, DDS_SampleLostStatus status)

on_sample_lost() is 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.

3.7.2.6 void(* DDS_DataReaderListener::on_sample_rejected)(DDS_DataReader the_reader, DDS_SampleRejectedStatus status)

on_sample_rejected() is 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.

3.7.2.7 void(* DDS_DataReaderListener::on_subscription_matched)(DDS_DataReader the_reader, DDS_SubscriptionMatchedStatus status)

on_subscription_matched() is 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 struct was generated from the following file:

- dds.h

3.8 DDS_DataReaderListener_cd Struct Reference

The DDS_DataReaderListener_cd provides asynchronous notification of DDS_DataReader events with additional callback data.

Public Attributes

- void(∗ on_requested_deadline_missed)(struct DDS_DataReaderListener_cd ∗self, DDS_DataReader the_reader, DDS_RequestedDeadlineMissedStatus status, void ∗callback_data)
- void(∗ on_requested_incompatible_qos)(struct DDS_DataReaderListener_cd ∗self, DDS_DataReader the_reader, DDS_RequestedIncompatibleQosStatus status, void ∗callback_data)
- void(∗ on_sample_rejected)(struct DDS_DataReaderListener_cd ∗self, DDS_DataReader the_reader, DDS_SampleRejectedStatus status, void ∗callback_data)
- void(∗ on_liveliness_changed)(struct DDS_DataReaderListener_cd ∗self, DDS_DataReader the_reader, DDS_LivelinessChangedStatus status, void ∗callback_data)
- void(∗ on_data_available)(struct DDS_DataReaderListener_cd ∗self, DDS_DataReader the_reader, void ∗callback_data)
- void(∗ on_subscription_matched)(struct DDS_DataReaderListener_cd ∗self, DDS_DataReader the_reader, DDS_SubscriptionMatchedStatus status, void ∗callback_data)
- void(∗ on_sample_lost)(struct DDS_DataReaderListener_cd ∗self, DDS_DataReader the_reader, DDS_SampleLostStatus status, void ∗callback_data)

3.8.1 Detailed Description

The DDS_DataReaderListener_cd provides asynchronous notification of DDS_DataReader events with additional callback data. This listener can be installed by calling DDS_DataReader_set_listener_cd().

Note

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

3.8.2 Member Data Documentation

3.8.2.1 void(∗ DDS_DataReaderListener_cd::on_data_available)(struct DDS_DataReaderListener_cd ∗self, DDS_DataReader the_reader, void ∗callback_data)

on_data_available() is 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.

3.8.2.2 void(* DDS_DataReaderListener_cd::on_liveliness_changed)(struct DDS_DataReaderListener_cd *self, DDS_DataReader the_reader, DDS_LivelinessChangedStatus status, void *callback_data)

on_liveliness_changed() is 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.

3.8.2.3 void(* DDS_DataReaderListener_cd::on_requested_deadline_missed)(struct DDS_DataReaderListener_cd *self, DDS_DataReader the_reader, DDS_RequestedDeadlineMissedStatus status, void *callback_data)

on_requested_deadline_missed() is 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.

3.8.2.4 void(* DDS_DataReaderListener_cd::on_requested_incompatible_qos)(struct DDS_DataReaderListener_cd *self, DDS_DataReader the_reader, DDS_RequestedIncompatibleQosStatus status, void *callback_data)

on_requested_incompatible_qos() is 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.

3.8.2.5 void(* DDS_DataReaderListener_cd::on_sample_lost)(struct DDS_DataReaderListener_cd *self, DDS_DataReader the_reader, DDS_SampleLostStatus status, void *callback_data)

on_sample_lost() is 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.

3.8.2.6 void(* DDS_DataReaderListener_cd::on_sample_rejected)(struct DDS_DataReaderListener_cd *self, DDS_DataReader the_reader, DDS_SampleRejectedStatus status, void *callback_data)

on_sample_rejected() is 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.

3.8.2.7 void(∗ DDS_DataReaderListener_cd::on_subscription_matched)(struct DDS_DataReaderListener_cd ∗self, DDS_DataReader the_reader, DDS_SubscriptionMatchedStatus status, void ∗callback_data)

on_subscription_matched() is 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 struct was generated from the following file:

- dds.h

3.9 DDS_DataReaderQos Struct Reference

Structure that holds DDS_DataReader Quality of Service policies.

Public Attributes

- DDS_DurabilityQosPolicy durability
  The durability policy requested by the DataReader.
- DDS_DeadlineQosPolicy deadline
  The requested update frequency for data instances.
- DDS_LatencyBudgetQosPolicy latency_budget
  The latency requested by the DataReader.
- DDS_LivelinessQosPolicy liveliness
  The liveliness mechanism requested by the DataReader.
- DDS_ReliabilityQosPolicy reliability
  The transport reliability requested by the DataReader.
- DDS_DestinationOrderQosPolicy destination_order
  The destination order logic requested by the DataReader.
- DDS_HistoryQosPolicy history
  The data history requested by the DataReader.
- DDS_ResourceLimitsQosPolicy resource_limits
  The resource limits set on the DataReader.
- DDS_UserDataQosPolicy user_data
  A sequence of octets associated with the DataReader.
- DDS_OwnershipQosPolicy ownership
  The type of 'ownership' offered by the DataReader.
- DDS_TimeBasedFilterQosPolicy time_based_filter
  The maximum update frequency required/desired by the DataReader.
- DDS_ReaderDataLifecycleQosPolicy reader_data_lifecycle
  Controls the auto-purge behavior of the DataReader.

3.9.1 Detailed Description

Structure that holds DDS_DataReader Quality of Service policies.

See also

- DDS_DataReader_set_qos()
- DDS_DataReader_get_qos()
- DDS_Subscriber_create_datareader()
- DDS_Subscriber_set_default_datareader_qos()
- DDS_Subscriber_get_default_datareader_qos()

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

- dds.h

3.10 DDS_DataWriter Struct Reference

The DDS_DataWriter entity provides an interface for the application to publish (write) data.

Related Functions

(Note that these are not member functions.)

- DDS_ReturnCode_t DDS_DataWriter_enable (DDS_DataWriter dw)
  Enables the DDS_DataWriter.
- DDS_InstanceHandle_t DDS_DataWriter_get_instance_handle (DDS_DataWriter dw)
  This operation returns the InstanceHandle_t that identifies the DataWriter.
- DDS_StatusCondition DDS_DataWriter_get_statuscondition (DDS_DataWriter dw)
  This operation allows access to the DDS_StatusCondition associated with the DataWriter.
- DDS_StatusMask DDS_DataWriter_get_status_changes (DDS_DataWriter dw)
  *This returns the list of **triggered** communication statuses in the DataWriter.*
- DDS_ReturnCode_t DDS_DataWriter_set_qos (DDS_DataWriter dw, const DDS_DataWriterQos *qos)
  Sets the DDS_DataWriterQos values.
- DDS_ReturnCode_t DDS_DataWriter_get_qos (DDS_DataWriter dw, DDS_DataWriterQos *qos)
  *Returns the current DDS_DataWriterQos settings held in the DataWriter **dw**.*
- DDS_ReturnCode_t DDS_DataWriter_set_listener (DDS_DataWriter dw, DDS_DataWriterListener *a_listener, DDS_StatusMask mask)
  *Installs a DDS_DataWriterListener on DataWriter **dw**.*
- DDS_ReturnCode_t DDS_DataWriter_set_listener_cd (DDS_DataWriter dw, DDS_DataWriterListener_cd *a_listener, DDS_StatusMask mask, void *callback_data)
  *Installs a DDS_DataWriterListener_cd on DataWriter **dw**.*
- DDS_DataWriterListener * DDS_DataWriter_get_listener (DDS_DataWriter dw)
  This operation returns the currently installed DDS_DataWriterListener.
- DDS_DataWriterListener_cd * DDS_DataWriter_get_listener_cd (DDS_DataWriter dw)
  This operation returns the currently installed DDS_DataWriterListener_cd.
- DDS_DataWriter_wait_for_acknowledgements

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

- DDS_Topic DDS_DataWriter_get_topic (DDS_DataWriter dw)

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

- DDS_Publisher DDS_DataWriter_get_publisher (DDS_DataWriter dw)

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

- DDS_ReturnCode_t DDS_DataWriter_assert_liveliness (DDS_DataWriter dw)

  *This operation manually asserts the liveliness of the DataWriter **dw**.*

- DDS_ReturnCode_t DDS_DataWriter_get_matched_subscriptions (DDS_DataWriter dw, DDS_InstanceHandleSeq *subscription_handles)

  *This operation retrieves the list of DataReaders currently matched with the DataWriter **dw**.*

- DDS_ReturnCode_t DDS_DataWriter_get_matched_subscription_data (DDS_DataWriter dw, DDS_SubscriptionBuiltinTopicData *subscription_data, const DDS_InstanceHandle_t subscription_handle)

  *This operation returns data that describes a particular matched DataReader identified by **subscription_handle**.*

- DDS_ReturnCode_t DDS_DataWriter_get_liveliness_lost_status (DDS_DataWriter dw, DDS_LivelinessLostStatus *status)

  Provides access to the current DDS_LivelinessLostStatus of the DataWriter.

- DDS_ReturnCode_t DDS_DataWriter_get_offered_deadline_missed_status (DDS_DataWriter dw, DDS_OfferedDeadlineMissedStatus *status)

  Provides access to the current DDS_OfferedDeadlineMissedStatus of the DataWriter.

- DDS_ReturnCode_t DDS_DataWriter_get_offered_incompatible_qos_status (DDS_DataWriter dw, DDS_OfferedIncompatibleQosStatus *status)

  Provides access to the current DDS_OfferedIncompatibleQosStatus of the DataWriter.

- DDS_ReturnCode_t DDS_DataWriter_get_publication_matched_status (DDS_DataWriter dw, DDS_PublicationMatchedStatus *status)

  Provides access to the current DDS_PublicationMatchedStatus of the DataWriter.

- DDS_ReturnCode_t DDS_DataWriter_get_key_value (DDS_DataWriter dw, void *key_holder, const DDS_InstanceHandle_t handle)

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

- DDS_InstanceHandle_t DDS_DataWriter_lookup_instance (DDS_DataWriter dw, const void *instance_data)

*Returns the handle that identifies the data instance provided in **instance_data**.*

- DDS_InstanceHandle_t DDS_DataWriter_register_instance (DDS_DataWriter dw, const void *instance_data)

  *Declares the existance of an instance identified by the 'key fields' in **instance_data**.*

- DDS_InstanceHandle_t DDS_DataWriter_register_instance_w_timestamp (DDS_DataWriter dw, const void *instance_data, const DDS_Time_t source_timestamp)

  *Declares the existance of an instance identified by the 'key fields' in **instance_data**.*

- DDS_ReturnCode_t DDS_DataWriter_unregister_instance (DDS_DataWriter dw, const void *instance_data, const DDS_InstanceHandle_t handle)

  Indicates that the writer will no longer be providing updates the specified instance.

- DDS_ReturnCode_t DDS_DataWriter_unregister_instance_w_timestamp (DDS_DataWriter dw, const void *instance_data, DDS_InstanceHandle_t handle, const DDS_Time_t source_timestamp)

  Indicates that the writer will no longer be providing updates the specified instance.

- DDS_ReturnCode_t DDS_DataWriter_write (DDS_DataWriter dw, const void *instance_data, const DDS_InstanceHandle_t handle)

  *Publishes the provided **instance_data**.*

- DDS_ReturnCode_t DDS_DataWriter_write_w_timestamp (DDS_DataWriter dw, const void *instance_data, const DDS_InstanceHandle_t handle, const DDS_Time_t source_timestamp)

  *Publishes the provided **instance_data**.*

- DDS_ReturnCode_t DDS_DataWriter_dispose (DDS_DataWriter dw, const void *instance_data, const DDS_InstanceHandle_t instance_handle)

  Indicates that the instance no longer exists.

- DDS_ReturnCode_t DDS_DataWriter_dispose_w_timestamp (DDS_DataWriter dw, const void *instance_data, const DDS_InstanceHandle_t instance_handle, const DDS_Time_t source_timestamp)

  Indicates that the instance no longer exists.

3.10.1 Detailed Description

The DDS_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.

3.10.2 Friends And Related Function Documentation

3.10.2.1 DDS_ReturnCode_t DDS_DataWriter_assert_liveliness (DDS_DataWriter *dw*) [related]

This operation manually asserts the liveliness of the DataWriter **dw**.

This operation is useful if the LIVELINESS QoS setting is MANUAL_BY_PARTICIPANT or MANUAL_-BY_TOPIC; otherwise, it has no effect.

Note

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.

3.10.2.2 DDS_ReturnCode_t DDS_DataWriter_dispose (DDS_DataWriter *dw,* const void * *instance_data,* const DDS_InstanceHandle_t *instance_handle*) [related]

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 timstamp.

3.10.2.3 DDS_ReturnCode_t DDS_DataWriter_dispose_w_timestamp (DDS_DataWriter *dw,* const void * *instance_data,* const DDS_InstanceHandle_t *instance_handle,* const DDS_Time_t *source_timestamp*) [related]

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.

3.10.2.4 DDS_ReturnCode_t DDS_DataWriter_enable (DDS_DataWriter *dw*) [related]

Enables the DDS_DataWriter.

A DataWriter is created either enabled or not based on the DDS_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 DDS_NOT_ENABLED error. DDS_DataWriter_enable() may be called on an already enabled DataWriter [it will have no effect].

3.10.2.5 DDS_ReturnCode_t DDS_DataWriter_get_key_value (DDS_DataWriter *dw,* void * *key_holder,* const DDS_InstanceHandle_t *handle*) [related]

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

Note

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.

3.10.2.6 DDS_DataWriterListener * DDS_DataWriter_get_listener (DDS_DataWriter *dw*) [related]

This operation returns the currently installed DDS_DataWriterListener.

Note

Because the infrastructure makes a copy of the listener provided in DDS_DataWriter_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.

3.10.2.7 DDS_DataWriterListener_cd * DDS_DataWriter_get_listener_cd (DDS_DataWriter *dw*) [related]

This operation returns the currently installed DDS_DataWriterListener_cd.

Note

Because the infrastructure holds a pointer to the listener provided in DDS_DataWriter_set_listener_cd(), the returned structure pointer will match the pointer originally provided. The application should not free the data referenced by the returned pointer.

3.10.2.8 DDS_ReturnCode_t DDS_DataWriter_get_liveliness_lost_status (DDS_DataWriter *dw,* DDS_LivelinessLostStatus * *status*) [related]

Provides access to the current DDS_LivelinessLostStatus of the DataWriter.

As a side-effect, this routine will reset the **total_count_change** status field to zero.

3.10.2.9 DDS_ReturnCode_t DDS_DataWriter_get_matched_subscription_data (DDS_DataWriter *dw,* DDS_SubscriptionBuiltinTopicData * *subscription_data,* const DDS_InstanceHandle_t *subscription_handle*) [related]

This operation returns data that describes a particular matched DataReader identified by **subscription_handle**.

An appropriate handle can be obtained through a call to DDS_DataWriter_get_matched_subscriptions().

If **subscription_handle** does not identify a matched DataReader, this routine will return DDS_RETCODE_PRECONDITION_NOT_MET.

3.10.2.10 DDS_ReturnCode_t DDS_DataWriter_get_matched_subscriptions (DDS_DataWriter *dw,* DDS_InstanceHandleSeq * *subscription_handles*) [related]

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 DDS_DomainParticipant_ignore_subscription(), then it will not appear in the list.

3.10.2.11 DDS_ReturnCode_t DDS_DataWriter_get_offered_deadline_missed_status (DDS_DataWriter *dw,* DDS_OfferedDeadlineMissedStatus * *status*) [related]

Provides access to the current DDS_OfferedDeadlineMissedStatus of the DataWriter.

As a side-effect, this routine will reset the **total_count_change** status field to zero.

3.10.2.12 DDS_ReturnCode_t DDS_DataWriter_get_offered_incompatible_qos_status (DDS_DataWriter *dw,* DDS_OfferedIncompatibleQosStatus * *status*) [related]

Provides access to the current DDS_OfferedIncompatibleQosStatus of the DataWriter.

As a side-effect, this routine will reset the **total_count_change** status field to zero.

3.10.2.13 DDS_ReturnCode_t DDS_DataWriter_get_publication_matched_status (DDS_DataWriter *dw,* DDS_PublicationMatchedStatus * *status*) [related]

Provides access to the current DDS_PublicationMatchedStatus of the DataWriter.

As a side-effect, this routine will reset the **total_count_change** and **current_count_change** status fields to zero.

3.10.2.14 DDS_ReturnCode_t DDS_DataWriter_get_qos (DDS_DataWriter *dw,* DDS_DataWriterQos * *qos*) [related]

Returns the current DDS_DataWriterQos settings held in the DataWriter **dw**.

This routines copies data from the DataWriter QoS properties into **qos**.

Note

The qos structure may contain sequences or strings that are populated with dynamic memory. The caller is responsible for freeing the dynamic memory of these items.
For example, the sequence 'qos->user_data' may have dynamically allocated memory assigned. This can be released by a call to **seq_clear(&qos->user_data)**.

3.10.2.15 DDS_StatusMask DDS_DataWriter_get_status_changes (DDS_DataWriter *dw*) [related]

This returns the list of **triggered** communication statuses in the DataWriter.

If the DataWriter is not enabled, all statuses will be **untriggered**.

3.10.2.16 DDS_StatusCondition DDS_DataWriter_get_statuscondition (DDS_DataWriter *dw*) [related]

This operation allows access to the DDS_StatusCondition associated with the DataWriter.

The returned condition can be added to a DDS_WaitSet.

3.10.2.17 DDS_InstanceHandle_t DDS_DataWriter_lookup_instance (DDS_DataWriter *dw,* const void * *instance_data*) [related]

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.

3.10.2.18 DDS_InstanceHandle_t DDS_DataWriter_register_instance (DDS_DataWriter *dw,* const void * *instance_data*) [related]

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 timestamp.

3.10.2.19 DDS_InstanceHandle_t DDS_DataWriter_register_instance_w_timestamp (DDS_DataWriter *dw*, const void * *instance_data*, const DDS_Time_t *source_timestamp*) [related]

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 timestamp.

3.10.2.20 DDS_ReturnCode_t DDS_DataWriter_set_listener (DDS_DataWriter *dw*, DDS_DataWriterListener * *a_listener*, DDS_StatusMask *mask*) [related]

Installs a DDS_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.

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

3.10.2.21 DDS_ReturnCode_t DDS_DataWriter_set_listener_cd (DDS_DataWriter *dw*, DDS_DataWriterListener_cd * *a_listener*, DDS_StatusMask *mask*, void * *callback_data*) [related]

Installs a DDS_DataWriterListener_cd on DataWriter **dw**.

Only one listener may be attached to a DataWriter at a time. A call to **set_listener_cd()** will replace any current listener with **a_listener**.

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

The infrastructure will hold a pointer to the listener structure which means that it must be persisted by the application.

3.10.2.22 DDS_ReturnCode_t DDS_DataWriter_set_qos (DDS_DataWriter *dw*, const DDS_DataWriterQos * *qos*) [related]

Sets the DDS_DataWriterQos values.

These QoS values affect the behavior of the DDS_DataWriter.

3.10.2.23 DDS_ReturnCode_t DDS_DataWriter_unregister_instance (DDS_DataWriter *dw*, const void * *instance_data*, const DDS_InstanceHandle_t *handle*) [related]

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 timstamp.

3.10.2.24 DDS_ReturnCode_t DDS_DataWriter_unregister_instance_w_timestamp (DDS_DataWriter *dw,* const void * *instance_data,* DDS_InstanceHandle_t *handle,* const DDS_Time_t *source_timestamp*) [related]

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 timstamp.

3.10.2.25 DDS_DataWriter_wait_for_acknowledgements [related]

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'

3.10.2.26 DDS_ReturnCode_t DDS_DataWriter_write (DDS_DataWriter *dw,* const void * *instance_data,* const DDS_InstanceHandle_t *handle*) [related]

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 DDS_RETCODE_TIMEOUT is returned.

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

On success, DDS_RETCODE_OK is returned.

3.10.2.27 DDS_ReturnCode_t DDS_DataWriter_write_w_timestamp (DDS_DataWriter *dw,* const void * *instance_data,* const DDS_InstanceHandle_t *handle,* const DDS_Time_t *source_timestamp*) [related]

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 DDS_RETCODE_TIMEOUT is returned.

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

On success, DDS_RETCODE_OK is returned.

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

- dds.h

3.11 DDS_DataWriterListener Struct Reference

The DDS_DataWriterListener provides asynchronous notification of DDS_DataWriter events.

Public Attributes

- void(∗ on_offered_deadline_missed)(DDS_DataWriter writer, DDS_OfferedDeadlineMissedStatus status)
- void(∗ on_offered_incompatible_qos)(DDS_DataWriter writer, DDS_OfferedIncompatibleQosStatus status)
- void(∗ on_liveliness_lost)(DDS_DataWriter writer, DDS_LivelinessLostStatus status)
- void(∗ on_publication_matched)(DDS_DataWriter writer, DDS_PublicationMatchedStatus status)

3.11.1 Detailed Description

The DDS_DataWriterListener provides asynchronous notification of DDS_DataWriter events. This listener can be installed during DataWriter creation, DDS_Publisher_create_datawriter(), as well as by calling DDS_-DataWriter_set_listener().

Note

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

3.11.2 Member Data Documentation

3.11.2.1 void(∗ DDS_DataWriterListener::on_liveliness_lost)(DDS_DataWriter writer, DDS_LivelinessLostStatus status)

on_liveliness_lost() is 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.

3.11.2.2 void(∗ DDS_DataWriterListener::on_offered_deadline_missed)(DDS_DataWriter writer, DDS_OfferedDeadlineMissedStatus status)

on_offered_deadline_missed() is 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.

3.11.2.3 void(* DDS_DataWriterListener::on_offered_incompatible_qos)(DDS_DataWriter writer, DDS_OfferedIncompatibleQosStatus status)

on_offered_incompatible_qos() is 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.

3.11.2.4 void(* DDS_DataWriterListener::on_publication_matched)(DDS_DataWriter writer, DDS_PublicationMatchedStatus status)

on_publication_matched() is 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 struct was generated from the following file:

- dds.h

3.12 DDS_DataWriterListener_cd Struct Reference

The DDS_DataWriterListener_cd provides asynchronous notification of DDS_DataWriter events with additional callback data.

Public Attributes

- void(∗ on_offered_deadline_missed)(struct DDS_DataWriterListener_cd ∗self, DDS_DataWriter writer, DDS_OfferedDeadlineMissedStatus status, void ∗callback_data)
- void(∗ on_offered_incompatible_qos)(struct DDS_DataWriterListener_cd ∗self, DDS_DataWriter writer, DDS_OfferedIncompatibleQosStatus status, void ∗callback_data)
- void(∗ on_liveliness_lost)(struct DDS_DataWriterListener_cd ∗self, DDS_DataWriter writer, DDS_LivelinessLostStatus status, void ∗callback_data)
- void(∗ on_publication_matched)(struct DDS_DataWriterListener_cd ∗self, DDS_DataWriter writer, DDS_PublicationMatchedStatus status, void ∗callback_data)

3.12.1 Detailed Description

The DDS_DataWriterListener_cd provides asynchronous notification of DDS_DataWriter events with additional callback data. This listener can be installed by calling DDS_DataWriter_set_listener_cd().

Note

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

3.12.2 Member Data Documentation

3.12.2.1 void(∗ DDS_DataWriterListener_cd::on_liveliness_lost)(struct DDS_DataWriterListener_cd ∗self, DDS_DataWriter writer, DDS_LivelinessLostStatus status, void ∗callback_data)

on_liveliness_lost() is 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.

3.12.2.2 void(∗ DDS_DataWriterListener_cd::on_offered_deadline_missed)(struct DDS_DataWriterListener_cd ∗self, DDS_DataWriter writer, DDS_OfferedDeadlineMissedStatus status, void ∗callback_data)

on_offered_deadline_missed() is 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.

3.12.2.3 void(* DDS_DataWriterListener_cd::on_offered_incompatible_qos)(struct DDS_DataWriterListener_cd *self, DDS_DataWriter writer, DDS_OfferedIncompatibleQosStatus status, void *callback_data)

on_offered_incompatible_qos() is 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.

3.12.2.4 void(* DDS_DataWriterListener_cd::on_publication_matched)(struct DDS_DataWriterListener_cd *self, DDS_DataWriter writer, DDS_PublicationMatchedStatus status, void *callback_data)

on_publication_matched() is 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 struct was generated from the following file:

- dds.h

3.13 DDS_DataWriterQos Struct Reference

Structure that holds DDS_DataWriter Quality of Service policies.

Public Attributes

- DDS_DurabilityQosPolicy durability
  The durability policy offered by the DataWriter.
- DDS_DurabilityServiceQosPolicy durability_service
  The durability service configuration offered by the DataWriter.
- DDS_DeadlineQosPolicy deadline
  The deadline committed to by the DataWriter.
- DDS_LatencyBudgetQosPolicy latency_budget
  The latency allowed by the DataWriter.
- DDS_LivelinessQosPolicy liveliness
  The liveliness mechanism offered by the DataWriter.
- DDS_ReliabilityQosPolicy reliability
  The transport reliability offered by the DataWriter.
- DDS_DestinationOrderQosPolicy destination_order
  The destination order logic offered by the DataWriter.
- DDS_HistoryQosPolicy history
  The data history maintained by the DataWriter.
- DDS_ResourceLimitsQosPolicy resource_limits
  The resource limits set on the DataWriter.
- DDS_TransportPriorityQosPolicy transport_priority
  The transport priority supported by the DataWriter.
- DDS_LifespanQosPolicy lifespan
  The expiration time for old samples managed by the DataWriter.
- DDS_UserDataQosPolicy user_data
  A sequence of octets associated with the DataWriter.

- DDS_OwnershipQosPolicy ownership

  The type of 'ownership' offered by the DataWriter.

- DDS_OwnershipStrengthQosPolicy ownership_strength

  The measure of 'ownership strength' offered by the DataWriter.

- DDS_WriterDataLifecycleQosPolicy writer_data_lifecycle

  Indicates if unregistered instances should be automatically disposed by the DataWriter.

3.13.1 Detailed Description

Structure that holds DDS_DataWriter Quality of Service policies.

See also

- DDS_DataWriter_set_qos()
- DDS_DataWriter_get_qos()
- DDS_Publisher_create_datawriter()
- DDS_Publisher_set_default_datawriter_qos()
- DDS_Publisher_get_default_datawriter_qos()

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

- dds.h

3.14 DDS_DomainParticipant Struct Reference

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

Related Functions

(Note that these are not member functions.)

- DDS_ReturnCode_t DDS_DomainParticipant_enable (DDS_DomainParticipant dp)

  Enables the DomainParticipant.

- DDS_InstanceHandle_t DDS_DomainParticipant_get_instance_handle (DDS_DomainParticipant dp)

  This operation returns the InstanceHandle_t that identifies the DomainParticipant.

- DDS_StatusCondition DDS_DomainParticipant_get_statuscondition (DDS_DomainParticipant dp)

  This operation allows access to the DDS_StatusCondition associated with the DomainParticipant.

- DDS_StatusMask DDS_DomainParticipant_get_status_changes (DDS_DomainParticipant dp)

  *This returns the list of **triggered** communication statuses in the DomainParticipant.*

- DDS_Publisher DDS_DomainParticipant_create_publisher (DDS_DomainParticipant dp, DDS_PublisherQos *qos, DDS_PublisherListener *a_listener, DDS_StatusMask mask)

  This operation creates a DDS_Publisher with the specified DDS_PublisherQoS settings and DDS_PublisherListener.

- DDS_ReturnCode_t DDS_DomainParticipant_delete_publisher (DDS_DomainParticipant dp, DDS_Publisher p)

  This operation deletes an existing DDS_Publisher.

- DDS_Subscriber DDS_DomainParticipant_create_subscriber (DDS_DomainParticipant dp, DDS_SubscriberQos *qos, DDS_SubscriberListener *a_listener, DDS_StatusMask mask)

  This operation creates a DDS_Subscriber with the specified DDS_SubscriberQos and DDS_SubscriberListener.

- DDS_ReturnCode_t DDS_DomainParticipant_delete_subscriber (DDS_DomainParticipant dp, DDS_Subscriber s)

  This operation deletes an existing DDS_Subscriber.

- DDS_Subscriber DDS_DomainParticipant_get_builtin_subscriber (struct _DomainParticipant *d)

  Returns the built-in DDS_Subscriber.

- DDS_Topic DDS_DomainParticipant_create_topic (DDS_DomainParticipant dp, const char *topic_name, const char *type_name, DDS_TopicQos *qos, DDS_TopicListener *a_listener, DDS_StatusMask mask)

  This operation creates a DDS_Topic with the specified DDS_TopicQos settings and DDS_TopicListener.

- DDS_ReturnCode_t DDS_DomainParticipant_delete_topic (DDS_DomainParticipant dp, DDS_Topic a_topic)

  This operation deletes a DDS_Topic.

- DDS_Topic DDS_DomainParticipant_find_topic (DDS_DomainParticipant dp, const char *topic_name, DDS_Duration_t *timeout)

  *This operation returns a DDS_Topic identified by the provided **topic_name**.*

- DDS_TopicDescription DDS_DomainParticipant_lookup_topicdescription (DDS_DomainParticipant dp, const char *name)

  *This operation returns an existing, locally-created DDS_TopicDescription, named **name**.*

- DDS_ContentFilteredTopic DDS_DomainParticipant_create_contentfilteredtopic (DDS_DomainParticipant dp, const char *name, const DDS_Topic related_topic, const char *filter_expression, const DDS_StringSeq *filter_parameters)

  This operation creates a ContentFilteredTopic.

- DDS_ReturnCode_t DDS_DomainParticipant_delete_contentfilteredtopic (DDS_DomainParticipant dp, DDS_ContentFilteredTopic a_contentfilteredtopic)

  This operation deletes a previously allocated ContentFilteredTopic.

- DDS_MultiTopic DDS_DomainParticipant_create_multitopic (DDS_DomainParticipant dp, const char *name, const char *type_name, const char *subscription_expression, const DDS_StringSeq *expression_parameters)
- DDS_ReturnCode_t DDS_DomainParticipant_delete_multitopic (DDS_DomainParticipant dp, DDS_MultiTopic a_multitopic)
- DDS_ReturnCode_t DDS_DomainParticipant_delete_contained_entities (DDS_DomainParticipant dp)

  *This operation deletes all the objects created by means of the **create** operations on DomainParticipant **dp**.*

- DDS_ReturnCode_t DDS_DomainParticipant_set_qos (DDS_DomainParticipant dp, const DDS_DomainParticipantQos *qos)

  Sets the DDS_DomainParticipantQoS values.

- DDS_ReturnCode_t DDS_DomainParticipant_get_qos (DDS_DomainParticipant dp, DDS_DomainParticipantQos *qos)

  Provides access to the DDS_DomainParticipantQoS settings of the DomainParticipant.

- DDS_ReturnCode_t DDS_DomainParticipant_set_listener (DDS_DomainParticipant dp, DDS_DomainParticipantListener *a_listener, DDS_StatusMask mask)

  This operation installs a DDS_DomainParticipantListener on the DomainParticipant.

- DDS_ReturnCode_t DDS_DomainParticipant_set_listener_cd (DDS_DomainParticipant dp, DDS_DomainParticipantListener_cd *a_listener, DDS_StatusMask mask, void *callback_data)

  This operation installs a DDS_DomainParticipantListener_cd on the DomainParticipant.

- DDS_DomainParticipantListener * DDS_DomainParticipant_get_listener (DDS_DomainParticipant dp)

  This operation returns the currently installed DDS_DomainParticipantListener.

- DDS_DomainParticipantListener_cd * DDS_DomainParticipant_get_listener_cd (DDS_DomainParticipant dp)

  This operation returns the currently installed DDS_DomainParticipantListener_cd.

- DDS_ReturnCode_t DDS_DomainParticipant_ignore_participant (DDS_DomainParticipant dp, const DDS_InstanceHandle_t handle)

  *Instructs the DomainParticipant **dp** to ignore the external DomainParticipant identified by **handle**.*

- DDS_ReturnCode_t DDS_DomainParticipant_ignore_topic (DDS_DomainParticipant dp, const DDS_InstanceHandle_t handle)

  *This operation instructs the DomainParticipant **dp** to ignore a Topic identified by **handle**.*

- DDS_ReturnCode_t DDS_DomainParticipant_ignore_publication (DDS_DomainParticipant dp, const DDS_InstanceHandle_t handle)

  *This operation instructs the DomainParticipant **dp** to ignore a Publication identified by **handle**.*

- DDS_ReturnCode_t DDS_DomainParticipant_ignore_subscription (DDS_DomainParticipant dp, const DDS_InstanceHandle_t handle)

  *This operation instructs the DomainParticipant **dp** to ignore a Subscription identified by **handle**.*

- DDS_DomainId_t DDS_DomainParticipant_get_domain_id (DDS_DomainParticipant dp)

  This operation retrieves the domain_id to which the DomainParticipant belongs.

- DDS_ReturnCode_t DDS_DomainParticipant_assert_liveliness (DDS_DomainParticipant dp)

  *This operation manually asserts the liveliness of the DomainParticipant **dp**.*

- DDS_ReturnCode_t DDS_DomainParticipant_set_default_publisher_qos (DDS_DomainParticipant d, DDS_PublisherQos *qos)

  Sets the default DDS_PublisherQos held in the DomainParticipant.

- DDS_ReturnCode_t DDS_DomainParticipant_get_default_publisher_qos (DDS_DomainParticipant d, DDS_PublisherQos *qos)

  Provides access to the default DDS_PublisherQos settings held in the factory.

- DDS_ReturnCode_t DDS_DomainParticipant_set_default_subscriber_qos (DDS_DomainParticipant d, DDS_SubscriberQos *qos)

  Sets the default DDS_SubscriberQos held in the DomainParticipant.

- DDS_ReturnCode_t DDS_DomainParticipant_get_default_subscriber_qos (DDS_DomainParticipant d, DDS_SubscriberQos *qos)

  Provides access to the default DDS_SubscriberQos settings held in the factory.

- DDS_ReturnCode_t DDS_DomainParticipant_set_default_topic_qos (DDS_DomainParticipant d, DDS_TopicQos *qos)

  Sets the default DDS_TopicQos held in the DomainParticipant.

- DDS_ReturnCode_t DDS_DomainParticipant_get_default_topic_qos (DDS_DomainParticipant d, DDS_TopicQos *qos)

  Provides access to the default DDS_TopicQos settings held in the factory.

- DDS_ReturnCode_t DDS_DomainParticipant_get_discovered_participants (DDS_DomainParticipant d, DDS_InstanceHandleSeq *participant_handles)

  This operation returns the list of handles identifying the DDS_DomainParticipant objects that have been discovered.

- DDS_ReturnCode_t DDS_DomainParticipant_get_discovered_participant_data (DDS_DomainParticipant d, DDS_ParticipantBuiltinTopicData *participant_data, const DDS_InstanceHandle_t participant_handle)

  *This operation returns data that describes a particular discovered participant identified by **participant_handle**.*

- DDS_ReturnCode_t DDS_DomainParticipant_get_discovered_topics (DDS_DomainParticipant d, DDS_InstanceHandleSeq *topic_handles)

  This operation returns the list of handles identifying the DDS_Topic objects that have been discovered.

- DDS_ReturnCode_t DDS_DomainParticipant_get_discovered_topic_data (DDS_DomainParticipant d, DDS_TopicBuiltinTopicData *topic_data, const DDS_InstanceHandle_t topic_handle)

  *This operation returns data that describes a particular discovered topic identified by **topic_handle**.*

- unsigned char DDS_DomainParticipant_contains_entity (DDS_DomainParticipant d, const DDS_InstanceHandle_t a_handle)

  *This operation checks whether or not the given handle **a_handle** represents an object that was created by the DomainParticipant **d**.*

- DDS_ReturnCode_t DDS_DomainParticipant_get_current_time (DDS_DomainParticipant d, DDS_Time_t *current_time)

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

- DDS_ReturnCode_t DDS_DomainParticipant_register_type (DDS_DomainParticipant dp, DDS_TypeSupport ts, const char *type_name)

  Registers a TypeSupport with the Participant.

- int DDS_DomainParticipant_check_version (DDS_DomainParticipant dp, const char *major, const char *minor, const char *patch)

  Tests the provide version information against the version of the running CoreDX library.

- DDS_ReturnCode_t DDS_DomainParticipant_do_work (DDS_DomainParticipant dp, DDS_Duration_t *time_slice)

  Transfer control to the DomainParticipant so it can do background processing.

- DDS_ReturnCode_t DDS_DomainParticipant_builtin_wait_for_acknowledgments (DDS_DomainParticipant dp, DDS_Duration_t *max_wait)

  Ensure that local discovery information has been received by all known peers.

3.14.1 Detailed Description

The DDS_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 Friends And Related Function Documentation

3.14.2.1 DDS_ReturnCode_t DDS_DomainParticipant_assert_liveliness (DDS_DomainParticipant *dp*) [related]

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 DDS_MANUAL_BY_PARTICIPANT. In this case, the operation will assert the liveliness of those particular DataWriter objects.

Note

Writing data via the DataWriter **write** operation automatically asserts the liveliness of the DataWriter and its containing DomainParticipant. Therefore, the use of DDS_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 int DDS_DomainParticipant_check_version (DDS_DomainParticipant *dp,* const char * *major,* const char * *minor,* const char * *patch*) [related]

Tests the provide version information against the version of the running CoreDX library.

If the provided major and minor numbers match those of the running library, then the routine returns 0. If the provided numbers are larger than those of the running library, then the routine returns a positive number (indicating that the provided version is greater than the library version). Finaly, if the provied version is less than the library version, a negative number is returned. Currently, the 'patch' value is ignored.

3.14.2.3 unsigned char DDS_DomainParticipant_contains_entity (DDS_DomainParticipant *d,* const DDS_InstanceHandle_t *a_handle*) [related]

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.

3.14.2.4 DDS_ContentFilteredTopic DDS_DomainParticipant_create_contentfilteredtopic (DDS_DomainParticipant *dp,* const char * *name,* const DDS_Topic *related_topic,* const char * *filter_expression,* const DDS_StringSeq * *filter_parameters*) [related]

This operation creates a ContentFilteredTopic.

The ContentFilteredTopic is associated with another un-filtered topic **related_topic**. It applies a filter to the data of the related topic. If a data sample passes the filter, it will be made available to a DataReader associated with the ContentFilteredTopic.

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

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

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

The filter_expression can refer to parameters. The syntax for 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.

3.14.2.5 DDS_MultiTopic DDS_DomainParticipant_create_multitopic (DDS_DomainParticipant *dp*, const char * *name*, const char * *type_name*, const char * *subscription_expression*, const DDS_StringSeq * *expression_parameters*) [related]

Not Yet Supported

This operation is not implemented yet.

3.14.2.6 DDS_Publisher DDS_DomainParticipant_create_publisher (DDS_DomainParticipant *dp*, DDS_PublisherQos * *qos*, DDS_PublisherListener * *a_listener*, DDS_StatusMask *mask*) [related]

This operation creates a DDS_Publisher with the specified DDS_PublisherQoS settings and DDS_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.

3.14.2.7 DDS_Subscriber DDS_DomainParticipant_create_subscriber (DDS_DomainParticipant *dp*, DDS_SubscriberQos * *qos*, DDS_SubscriberListener * *a_listener*, DDS_StatusMask *mask*) [related]

This operation creates a DDS_Subscriber with the specified DDS_SubscriberQos and DDS_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.

3.14.2.8 DDS_Topic DDS_DomainParticipant_create_topic (DDS_DomainParticipant *dp*, const char * *topic_name*, const char * *type_name*, DDS_TopicQos * *qos*, DDS_TopicListener * *a_listener*, DDS_StatusMask *mask*) [related]

This operation creates a DDS_Topic with the specified DDS_TopicQos settings and DDS_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 DDS_DomainParticipant_-delete_topic().

3.14.2.9 DDS_ReturnCode_t DDS_DomainParticipant_delete_contained_entities (DDS_DomainParticipant *dp*) [related]

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 DDS_DomainParticipantFactory_delete_-participant().

If any of the objects cannot be deleted, this routine will return DDS_RETCODE_PRECONDITION_NOT_-MET.

3.14.2.10 DDS_ReturnCode_t DDS_DomainParticipant_delete_contentfilteredtopic (DDS_DomainParticipant *dp,* DDS_ContentFilteredTopic *a_contentfilteredtopic*) [related]

This operation deletes a previously allocated ContentFilteredTopic.

This operation will fail if there are any DDS_DataReader, DDS_DataWriter, DDS_ContentFilteredTopic, or DDS_MultiTopic objects that reference the specified Topic. In this case, the operation will return DDS_-RETCODE_PRECONDITION_NOT_MET.

The Topic must be owned by DomainPartipant **dp**; otherwise DDS_RETCODE_PRECONDITION_NOT_-MET will be returned.

3.14.2.11 DDS_ReturnCode_t DDS_DomainParticipant_delete_multitopic (DDS_DomainParticipant *dp,* DDS_MultiTopic *a_multitopic*) [related]

Not Yet Supported

This operation is not implemented yet.

3.14.2.12 DDS_ReturnCode_t DDS_DomainParticipant_delete_publisher (DDS_DomainParticipant *dp,* DDS_Publisher *p*) [related]

This operation deletes an existing DDS_Publisher.

A Publisher cannot be deleted if it contains any DDS_DataWriter objects. In this case, **delete_publisher** will return DDS_RETCODE_PRECONDITION_NOT_MET.

If the DomainParticipant **dp** does not contain the provided Publisher **p**, the operation will return **DDS_-RETCODE_PRECONDITION_NOT_MET**.

3.14.2.13 DDS_ReturnCode_t DDS_DomainParticipant_delete_subscriber (DDS_DomainParticipant *dp,* DDS_Subscriber *s*) [related]

This operation deletes an existing DDS_Subscriber.

A Subscriber cannot be deleted if it contains any DDS_DataReader objects. In this case, **delete_subscriber** will return DDS_RETCODE_PRECONDITION_NOT_MET.

If the DomainParticipant **dp** does not contain the provided Subscriber **s**, the operation will return **DDS_-RETCODE_PRECONDITION_NOT_MET**.

3.14.2.14 DDS_ReturnCode_t DDS_DomainParticipant_delete_topic (DDS_DomainParticipant *dp,* DDS_Topic *a_topic*) [related]

This operation deletes a DDS_Topic.

This operation will fail if there are any DDS_DataReader, DDS_DataWriter, DDS_ContentFilteredTopic, or DDS_MultiTopic objects that reference the specified Topic. In this case, the operation will return DDS_-RETCODE_PRECONDITION_NOT_MET.

The Topic must be owned by DomainPartipant **dp**; otherwise DDS_RETCODE_PRECONDITION_NOT_-MET will be returned.

3.14.2.15 DDS_ReturnCode_t DDS_DomainParticipant_do_work (DDS_DomainParticipant *dp,* DDS_Duration_t * *time_slice*) [related]

Transfer control to the DomainParticipant so it can do background processing.

This function is to be called only if the DomainParticipant is configured to use NO threads via the thread_-model QoS policy. The participant will continue to execute background processing until at least 'time_slice' time passes. This routine will attempt to use no more than 'time_slice' time, but this is not a guarantee.

Note

If the DomainParticipant has threads (default case), then the Participant takes care of scheduling background work on its own, and this routine is not necessary and should not be called.

3.14.2.16 DDS_ReturnCode_t DDS_DomainParticipant_enable (DDS_DomainParticipant *dp*) [related]

Enables the DomainParticipant.

A DomainParticipant is created either enabled or not based on the DDS_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 DDS_NOT_ENABLED error. DDS_DomainParticipant_enable() may be called on an already enabled DomainParticipant [it will have no effect].

3.14.2.17 DDS_Topic DDS_DomainParticipant_find_topic (DDS_DomainParticipant *dp,* const char * *topic_name,* DDS_Duration_t * *timeout*) [related]

This operation returns a DDS_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 DDS_DomainParticipant_create_topic(), the topic returned from this operation (if not NULL) must be deleted by calling DDS_DomainParticipant_delete_topic().

3.14.2.18 DDS_Subscriber DDS_DomainParticipant_get_builtin_subscriber (struct _DomainParticipant * *d*) [related]

Returns the built-in DDS_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 DDS_DomainParticipant, DDS_Topic, DDS_DataReader, and DDS_DataWriter objects.

An application should not explicitly delete the Subscriber returned by this operation - it is managed internally.

3.14.2.19 DDS_ReturnCode_t DDS_DomainParticipant_get_default_publisher_qos (DDS_DomainParticipant *d*, DDS_PublisherQos * *qos*) [related]

Provides access to the default DDS_PublisherQos settings held in the factory.

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

3.14.2.20 DDS_ReturnCode_t DDS_DomainParticipant_get_default_subscriber_qos (DDS_DomainParticipant *d*, DDS_SubscriberQos * *qos*) [related]

Provides access to the default DDS_SubscriberQos settings held in the factory.

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

3.14.2.21 DDS_ReturnCode_t DDS_DomainParticipant_get_default_topic_qos (DDS_DomainParticipant *d*, DDS_TopicQos * *qos*) [related]

Provides access to the default DDS_TopicQos settings held in the factory.

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

3.14.2.22 DDS_ReturnCode_t DDS_DomainParticipant_get_discovered_participant_data (DDS_DomainParticipant *d*, DDS_ParticipantBuiltinTopicData * *participant_data*, const DDS_InstanceHandle_t *participant_handle*) [related]

This operation returns data that describes a particular discovered participant identified by **participant_handle**.

An appropriate handle can be obtained through a call to DDS_DomainParticipant_get_discovered_participants().

If **participant_handle** does not identify a known DomainParticipant, this routine will return DDS_RETCODE_PRECONDITION_NOT_MET.

3.14.2.23 DDS_ReturnCode_t DDS_DomainParticipant_get_discovered_participants (DDS_DomainParticipant *d*, DDS_InstanceHandleSeq * *participant_handles*) [related]

This operation returns the list of handles identifying the DDS_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 DDS_DomainParticipant_ignore_participant().

This routine allocates memory to populate the **participant_handles** sequence. The application is responsible for freeing this memory.

3.14.2.24 DDS_ReturnCode_t DDS_DomainParticipant_get_discovered_topic_data (DDS_DomainParticipant *d,* DDS_TopicBuiltinTopicData * *topic_data,* const DDS_InstanceHandle_t *topic_handle*) [related]

This operation returns data that describes a particular discovered topic identified by **topic_handle**.

An appropriate handle can be obtained through a call to DDS_DomainParticipant_get_discovered_topics().

If **topic_handle** does not identify a known Topic, this routine will return DDS_RETCODE_PRECONDITION_NOT_MET.

3.14.2.25 DDS_ReturnCode_t DDS_DomainParticipant_get_discovered_topics (DDS_DomainParticipant *d,* DDS_InstanceHandleSeq * *topic_handles*) [related]

This operation returns the list of handles identifying the DDS_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 DDS_DomainParticipant_ignore_topic().

This routine allocates memory to populate the **topic_handles** sequence. The application is responsible for freeing this memory.

3.14.2.26 DDS_DomainParticipantListener * DDS_DomainParticipant_get_listener (DDS_DomainParticipant *dp*) [related]

This operation returns the currently installed DDS_DomainParticipantListener.

Note

Because the infrastructure makes a copy of the listener provided in DDS_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 DDS_DomainParticipantListener_cd * DDS_DomainParticipant_get_listener_cd (DDS_DomainParticipant *dp*) [related]

This operation returns the currently installed DDS_DomainParticipantListener_cd.

Note

Because the infrastructure does not make a copy of the listener_cd provided in DDS_DomainParticipant_set_listener_cd(), the returned structure pointer will match the pointer originally provided. The application should not free the data referenced by the returned pointer.

3.14.2.28 DDS_StatusMask DDS_DomainParticipant_get_status_changes (DDS_DomainParticipant *dp*) [related]

This returns the list of **triggered** communication statuses in the DomainParticipant.

If the DomainParticipant is not enabled, all statuses will be **untriggered**. The list returned by **get_status_changes** may be empty. The list of statuses returned by **get_status_changes** operation contains statuses that are triggered on the DomainParticipant itself and does not include statuses from contained objects.

3.14.2.29 DDS_StatusCondition DDS_DomainParticipant_get_statuscondition (DDS_DomainParticipant *dp*) [related]

This operation allows access to the DDS_StatusCondition associated with the DomainParticipant.

The returned condition can be added to a DDS_WaitSet.

3.14.2.30 DDS_ReturnCode_t DDS_DomainParticipant_ignore_participant (DDS_DomainParticipant *dp*, const DDS_InstanceHandle_t *handle*) [related]

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.

3.14.2.31 DDS_ReturnCode_t DDS_DomainParticipant_ignore_publication (DDS_DomainParticipant *dp*, const DDS_InstanceHandle_t *handle*) [related]

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 operation is not implemented yet.

3.14.2.32 DDS_ReturnCode_t DDS_DomainParticipant_ignore_subscription (DDS_DomainParticipant *dp*, const DDS_InstanceHandle_t *handle*) [related]

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 operation is not implemented yet.

3.14.2.33 DDS_ReturnCode_t DDS_DomainParticipant_ignore_topic (DDS_DomainParticipant *dp,* const DDS_InstanceHandle_t *handle*) [related]

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 operation is not implemented yet.

3.14.2.34 DDS_TopicDescription DDS_DomainParticipant_lookup_topicdescription (DDS_DomainParticipant *dp,* const char * *name*) [related]

This operation returns an existing, locally-created DDS_TopicDescription, named **name**.

If a TopicDescription named **name** does not exist, then this routine will return NULL.

3.14.2.35 DDS_ReturnCode_t DDS_DomainParticipant_register_type (DDS_DomainParticipant *dp,* DDS_TypeSupport *ts,* const char * *type_name*) [related]

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, then an error is returned and the previously registered TypeSupport is maintained.

3.14.2.36 DDS_ReturnCode_t DDS_DomainParticipant_set_default_publisher_qos (DDS_DomainParticipant *d,* DDS_PublisherQos * *qos*) [related]

Sets the default DDS_PublisherQos held in the DomainParticipant.

This default qos will be used during subsequent calls to DDS_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, DDS_-INCONSISTENT_POLICY will be returned, and no changes will be made to the DomainParticipant.

3.14.2.37 DDS_ReturnCode_t DDS_DomainParticipant_set_default_subscriber_qos (DDS_DomainParticipant *d,* DDS_SubscriberQos * *qos*) [related]

Sets the default DDS_SubscriberQos held in the DomainParticipant.

This default qos will be used during subsequent calls to DDS_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, DDS_-INCONSISTENT_POLICY will be returned, and no changes will be made to the DomainParticipant.

3.14.2.38 DDS_ReturnCode_t DDS_DomainParticipant_set_default_topic_qos (DDS_DomainParticipant *d,* DDS_TopicQos * *qos*) [related]

Sets the default DDS_TopicQos held in the DomainParticipant.

This default qos will be used during subsequent calls to DDS_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, DDS_-INCONSISTENT_POLICY will be returned, and no changes will be made to the DomainParticipant.

3.14.2.39 DDS_ReturnCode_t DDS_DomainParticipant_set_listener (DDS_DomainParticipant *dp,* DDS_DomainParticipantListener * *a_listener,* DDS_StatusMask *mask*) [related]

This operation installs a DDS_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.

3.14.2.40 DDS_ReturnCode_t DDS_DomainParticipant_set_listener_cd (DDS_DomainParticipant *dp,* DDS_DomainParticipantListener_cd * *a_listener,* DDS_StatusMask *mask,* void * *callback_data*) [related]

This operation installs a DDS_DomainParticipantListener_cd on the DomainParticipant.

Only one listener may be attached to a DomainParticipant at a time. A call to **set_listener_cd()** will replace any current listener with **a_listener**.

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

The infrastructure does not make an internal copy of the listener structure which means that it must persisted by the application.

3.14.2.41 DDS_ReturnCode_t DDS_DomainParticipant_set_qos (DDS_DomainParticipant *dp*, const DDS_DomainParticipantQos * *qos*) [related]

Sets the DDS_DomainParticipantQoS values.

These QoS values affect the behavior of the DomainParticipant.

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

- dds.h

3.15 DDS_DomainParticipantFactory Struct Reference

The DDS_DomainParticipantFactory is used to configure, create and destroy DomainParticipant objects.

Related Functions

(Note that these are not member functions.)

- DDS_DomainParticipant DDS_DomainParticipantFactory_create_participant (DDS_DomainId_t domain_id, DDS_DomainParticipantQos *qos, DDS_DomainParticipantListener *a_listener, DDS_StatusMask mask)

  This operation creates a new DomainParticipant object.

- DDS_ReturnCode_t DDS_DomainParticipantFactory_set_license (const char *lic)

  Configures the CoreDX DDS run-time license.

- DDS_ReturnCode_t DDS_DomainParticipantFactory_delete_participant (DDS_DomainParticipant a_participant)

  Destroys the provided DomainParticipant.

- DDS_DomainParticipant DDS_DomainParticipantFactory_lookup_participant (DDS_DomainId_t domain_id)

  *Returns a previously created DomainParticipant belonging to the specified **domain_id**.*

- DDS_ReturnCode_t DDS_DomainParticipantFactory_set_default_participant_qos (DDS_DomainParticipantQos *qos)

  Sets the default DDS_DomainParticipantQos held in the factory.

- DDS_ReturnCode_t DDS_DomainParticipantFactory_get_default_participant_qos (DDS_DomainParticipantQos *qos)

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

- DDS_ReturnCode_t DDS_DomainParticipantFactory_set_qos (const DDS_DomainParticipantFactoryQos *qos)

  Sets the DomainParticipantFactory QoS values.

- DDS_ReturnCode_t DDS_DomainParticipantFactory_get_qos (DDS_DomainParticipantFactoryQos *qos)

  Provides access to the QoS settings of the DomainParticipantFactory.

3.15.1 Detailed Description

The DDS_DomainParticipantFactory is used to configure, create and destroy DomainParticipant objects.

3.15.2 Friends And Related Function Documentation

3.15.2.1 DDS_DomainParticipant DDS_DomainParticipantFactory_create_participant (DDS_DomainId_t *domain_id,* DDS_DomainParticipantQos * *qos,* DDS_DomainParticipantListener * *a_listener,* DDS_StatusMask *mask*) [related]

This operation creates a new DomainParticipant object.

The caller provides the **domain_id** to which the Participant should belong. The **listener** and **mask** arguments are used to specify a set of callback routines which will be invoked upon detection of certain events. The **qos** argument specifies the DomainParticipant Quality of Service settings that should be used when creating the DomainParticipant. It may be specified as **DDS_PARTICIPANT_QOS_DEFAULT** to instruct CoreDX to use the default qos settings held in the DomainParticipantFactory.

This routine will return **NULL** if it fails to create a DomainParticipant.

3.15.2.2 DDS_ReturnCode_t DDS_DomainParticipantFactory_delete_participant (DDS_DomainParticipant *a_participant*) [related]

Destroys the provided DomainParticipant.

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

3.15.2.3 DDS_ReturnCode_t DDS_DomainParticipantFactory_get_default_participant_qos (DDS_DomainParticipantQos * *qos*) [related]

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

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

3.15.2.4 DDS_DomainParticipant DDS_DomainParticipantFactory_lookup_participant (DDS_DomainId_t *domain_id*) [related]

Returns a previously created DomainParticipant belonging to the specified **domain_id**.

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

3.15.2.5 DDS_ReturnCode_t DDS_DomainParticipantFactory_set_default_participant_qos (DDS_DomainParticipantQos * *qos*) [related]

Sets the default DDS_DomainParticipantQos held in the factory.

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

This routine may fail if the provided **qos** argument is not internally consistent. In this case, DDS_-INCONSISTENT_POLICY will be returned, and no changes will be made to the DomainParticipantFactory.

3.15.2.6 DDS_ReturnCode_t DDS_DomainParticipantFactory_set_license (const char * *lic*) [related]

Configures the CoreDX DDS run-time license.

This routine will configure CoreDX DDS to utilize the provided license at run-time. The 'lic' parameter may be either: 1) the full path and filename of a file containing CoreDX DDS license(s); or 2) the complete license string surrounded by '<' and '>'.

3.15.2.7 DDS_ReturnCode_t DDS_DomainParticipantFactory_set_qos (const DDS_DomainParticipantFactoryQos * *qos*) [related]

Sets the DomainParticipantFactory QoS values.

These QoS values affect the behavior of the factory.

This routine may fail if the provided **qos** argument is not internally consistent. In this case, DDS_-INCONSISTENT_POLICY will be returned, and no changes will be made to the DomainParticipantFactory.

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

- dds.h

3.16 DDS_DomainParticipantFactoryQos Struct Reference

Structure that holds DDS_DomainParticipantFactory Quality of Service policies.

Public Attributes

- DDS_EntityFactoryQosPolicy entity_factory

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

3.16.1 Detailed Description

Structure that holds DDS_DomainParticipantFactory Quality of Service policies.

See also

- DDS_DomainParticipantFactory_set_qos()
- DDS_DomainParticipantFactory_get_qos()

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

- dds.h

3.17 DDS_DomainParticipantListener Struct Reference

The DDS_DomainParticipantListener provides asynchronous notification of DDS_DomainParticipant events.

Public Attributes

- void(∗ on_inconsistent_topic)(DDS_Topic the_topic, DDS_InconsistentTopicStatus status)
- void(∗ on_offered_deadline_missed)(DDS_DataWriter writer, DDS_OfferedDeadlineMissedStatus status)
- void(∗ on_offered_incompatible_qos)(DDS_DataWriter writer, DDS_OfferedIncompatibleQosStatus status)
- void(∗ on_liveliness_lost)(DDS_DataWriter writer, DDS_LivelinessLostStatus status)
- void(∗ on_publication_matched)(DDS_DataWriter writer, DDS_PublicationMatchedStatus status)
- void(∗ on_requested_deadline_missed)(DDS_DataReader the_reader, DDS_-RequestedDeadlineMissedStatus status)
- void(∗ on_requested_incompatible_qos)(DDS_DataReader the_reader, DDS_-RequestedIncompatibleQosStatus status)
- void(∗ on_sample_rejected)(DDS_DataReader the_reader, DDS_SampleRejectedStatus status)
- void(∗ on_liveliness_changed)(DDS_DataReader the_reader, DDS_LivelinessChangedStatus status)
- void(∗ on_data_available)(DDS_DataReader the_reader)
- void(∗ on_subscription_matched)(DDS_DataReader the_reader, DDS_SubscriptionMatchedStatus status)
- void(∗ on_sample_lost)(DDS_DataReader the_reader, DDS_SampleLostStatus status)
- void(∗ on_data_on_readers)(DDS_Subscriber the_subscriber)

3.17.1 Detailed Description

The DDS_DomainParticipantListener provides asynchronous notification of DDS_DomainParticipant events. This listener can be installed during DomainParticipant creation, DDS_DomainParticipantFactory_create_-participant(), as well as by calling DDS_DomainParticipant_set_listener().

Note

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

3.17.2 Member Data Documentation

3.17.2.1 void(∗ DDS_DomainParticipantListener::on_data_available)(DDS_DataReader the_reader)

on_data_available() is 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 DDS_DataReader and DDS_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.

3.17.2.2 void(* DDS_DomainParticipantListener::on_data_on_readers)(DDS_Subscriber the_subscriber)

on_data_on_readers() is 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 DDS_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.

3.17.2.3 void(* DDS_DomainParticipantListener::on_inconsistent_topic)(DDS_Topic the_topic, DDS_InconsistentTopicStatus status)

on_inconsistent_topic() is 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 DDS_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.

3.17.2.4 void(* DDS_DomainParticipantListener::on_liveliness_changed)(DDS_DataReader the_reader, DDS_LivelinessChangedStatus status)

on_liveliness_changed() is 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 DDS_DataReader and DDS_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.

3.17.2.5 void(* DDS_DomainParticipantListener::on_liveliness_lost)(DDS_DataWriter writer, DDS_LivelinessLostStatus status)

on_liveliness_lost() is 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 DDS_DataWriter and DDS_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.

3.17.2.6 void(* DDS_DomainParticipantListener::on_offered_deadline_missed)(DDS_DataWriter writer, DDS_OfferedDeadlineMissedStatus status)

on_offered_deadline_missed() is 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 DDS_DataWriter and DDS_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.

3.17.2.7 void(* DDS_DomainParticipantListener::on_offered_incompatible_qos)(DDS_DataWriter writer, DDS_OfferedIncompatibleQosStatus status)

on_offered_incompatible_qos() is 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 DDS_DataWriter and DDS_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.

3.17.2.8 void(* DDS_DomainParticipantListener::on_publication_matched)(DDS_DataWriter writer, DDS_PublicationMatchedStatus status)

on_publication_matched() is 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 DDS_DataWriter and DDS_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.

3.17.2.9 void(* DDS_DomainParticipantListener::on_requested_deadline_missed)(DDS_DataReader the_reader, DDS_RequestedDeadlineMissedStatus status)

on_requested_deadline_missed() is 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 DDS_DataReader and DDS_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.

3.17.2.10 void(* DDS_DomainParticipantListener::on_requested_incompatible_qos)(DDS_DataReader the_reader, DDS_RequestedIncompatibleQosStatus status)

on_requested_incompatible_qos() is 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 DDS_DataReader and DDS_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.

3.17.2.11 void(* DDS_DomainParticipantListener::on_sample_lost)(DDS_DataReader the_reader, DDS_SampleLostStatus status)

on_sample_lost() is called 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 DDS_DataReader and DDS_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.

3.17.2.12 void(* DDS_DomainParticipantListener::on_sample_rejected)(DDS_DataReader the_reader, DDS_SampleRejectedStatus status)

on_sample_rejected() is 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 DDS_DataReader and DDS_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.

3.17.2.13 void(* DDS_DomainParticipantListener::on_subscription_matched)(DDS_DataReader the_reader, DDS_SubscriptionMatchedStatus status)

on_subscription_matched() is 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 DDS_DataReader and DDS_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 struct was generated from the following file:

- dds.h

3.18 DDS_DomainParticipantListener_cd Struct Reference

The DDS_DomainParticipantListener_cd provides asynchronous notification of DDS_DomainParticipant events with additional callback data arguments.

Public Attributes

- void(* on_inconsistent_topic)(struct DDS_DomainParticipantListener_cd *self, DDS_Topic the_topic, DDS_InconsistentTopicStatus status, void *callback_data)
- void(* on_offered_deadline_missed)(struct DDS_DomainParticipantListener_cd *self, DDS_DataWriter writer, DDS_OfferedDeadlineMissedStatus status, void *callback_data)
- void(* on_offered_incompatible_qos)(struct DDS_DomainParticipantListener_cd *self, DDS_DataWriter writer, DDS_OfferedIncompatibleQosStatus status, void *callback_data)
- void(* on_liveliness_lost)(struct DDS_DomainParticipantListener_cd *self, DDS_DataWriter writer, DDS_LivelinessLostStatus status, void *callback_data)
- void(* on_publication_matched)(struct DDS_DomainParticipantListener_cd *self, DDS_DataWriter writer, DDS_PublicationMatchedStatus status, void *callback_data)
- void(* on_requested_deadline_missed)(struct DDS_DomainParticipantListener_cd *self, DDS_DataReader the_reader, DDS_RequestedDeadlineMissedStatus status, void *callback_data)
- void(* on_requested_incompatible_qos)(struct DDS_DomainParticipantListener_cd *self, DDS_DataReader the_reader, DDS_RequestedIncompatibleQosStatus status, void *callback_data)
- void(* on_sample_rejected)(struct DDS_DomainParticipantListener_cd *self, DDS_DataReader the_reader, DDS_SampleRejectedStatus status, void *callback_data)
- void(* on_liveliness_changed)(struct DDS_DomainParticipantListener_cd *self, DDS_DataReader the_reader, DDS_LivelinessChangedStatus status, void *callback_data)
- void(* on_data_available)(struct DDS_DomainParticipantListener_cd *self, DDS_DataReader the_reader, void *callback_data)
- void(* on_subscription_matched)(struct DDS_DomainParticipantListener_cd *self, DDS_DataReader the_reader, DDS_SubscriptionMatchedStatus status, void *callback_data)
- void(* on_sample_lost)(struct DDS_DomainParticipantListener_cd *self, DDS_DataReader the_reader, DDS_SampleLostStatus status, void *callback_data)
- void(* on_data_on_readers)(struct DDS_DomainParticipantListener_cd *self, DDS_Subscriber the_subscriber, void *callback_data)

3.18.1 Detailed Description

The DDS_DomainParticipantListener_cd provides asynchronous notification of DDS_DomainParticipant events with additional callback data arguments. This listener can be installed by calling DDS_DomainParticipant_set_listener_cd().

Note

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

3.18.2 Member Data Documentation

3.18.2.1 void(* DDS_DomainParticipantListener_cd::on_data_available)(struct DDS_DomainParticipantListener_cd *self, DDS_DataReader the_reader, void *callback_data)

on_data_available() is 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 DDS_DataReader and DDS_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.

3.18.2.2 void(* DDS_DomainParticipantListener_cd::on_data_on_readers)(struct DDS_DomainParticipantListener_cd *self, DDS_Subscriber the_subscriber, void *callback_data)

on_data_on_readers() is 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 DDS_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.

3.18.2.3 void(* DDS_DomainParticipantListener_cd::on_inconsistent_topic)(struct DDS_DomainParticipantListener_cd *self, DDS_Topic the_topic, DDS_InconsistentTopicStatus status, void *callback_data)

on_inconsistent_topic() is 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 DDS_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.

3.18.2.4 void(* DDS_DomainParticipantListener_cd::on_liveliness_changed)(struct DDS_DomainParticipantListener_cd *self, DDS_DataReader the_reader, DDS_LivelinessChangedStatus status, void *callback_data)

on_liveliness_changed() is 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 DDS_DataReader and DDS_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.

3.18.2.5 void(* DDS_DomainParticipantListener_cd::on_liveliness_lost)(struct DDS_DomainParticipantListener_cd *self, DDS_DataWriter writer, DDS_LivelinessLostStatus status, void *callback_data)

on_liveliness_lost() is 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 DDS_DataWriter and DDS_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.

3.18.2.6 void(* DDS_DomainParticipantListener_cd::on_offered_deadline_missed)(struct DDS_DomainParticipantListener_cd *self, DDS_DataWriter writer, DDS_OfferedDeadlineMissedStatus status, void *callback_data)

on_offered_deadline_missed() is 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 DDS_DataWriter and DDS_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.

3.18.2.7 void(* DDS_DomainParticipantListener_cd::on_offered_incompatible_qos)(struct DDS_DomainParticipantListener_cd *self, DDS_DataWriter writer, DDS_OfferedIncompatibleQosStatus status, void *callback_data)

on_offered_incompatible_qos() is 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 DDS_DataWriter and DDS_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.

3.18.2.8 void(* DDS_DomainParticipantListener_cd::on_publication_matched)(struct DDS_DomainParticipantListener_cd *self, DDS_DataWriter writer, DDS_PublicationMatchedStatus status, void *callback_data)

on_publication_matched() is 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 DDS_DataWriter and DDS_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.

3.18.2.9 void(* DDS_DomainParticipantListener_cd::on_requested_deadline_missed)(struct DDS_DomainParticipantListener_cd *self, DDS_DataReader the_reader, DDS_RequestedDeadlineMissedStatus status, void *callback_data)

on_requested_deadline_missed() is 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 DDS_DataReader and DDS_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.

3.18.2.10 void(* DDS_DomainParticipantListener_cd::on_requested_incompatible_qos)(struct DDS_DomainParticipantListener_cd *self, DDS_DataReader the_reader, DDS_RequestedIncompatibleQosStatus status, void *callback_data)

on_requested_incompatible_qos() is 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 DDS_DataReader and DDS_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.

3.18.2.11 void(* DDS_DomainParticipantListener_cd::on_sample_lost)(struct DDS_DomainParticipantListener_cd *self, DDS_DataReader the_reader, DDS_SampleLostStatus status, void *callback_data)

on_sample_lost() is called 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 DDS_DataReader and DDS_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.

3.18.2.12 void(* DDS_DomainParticipantListener_cd::on_sample_rejected)(struct DDS_DomainParticipantListener_cd *self, DDS_DataReader the_reader, DDS_SampleRejectedStatus status, void *callback_data)

on_sample_rejected() is 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 DDS_DataReader and DDS_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.

3.18.2.13 void(* DDS_DomainParticipantListener_cd::on_subscription_matched)(struct DDS_DomainParticipantListener_cd *self, DDS_DataReader the_reader, DDS_SubscriptionMatchedStatus status, void *callback_data)

on_subscription_matched() is 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 DDS_DataReader and DDS_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 struct was generated from the following file:

- dds.h

3.19 DDS_DomainParticipantQos Struct Reference

Structure that holds DDS_DomainParticipant Quality of Service policies.

Public Attributes

- DDS_UserDataQosPolicy user_data

  A sequence of octets associated with a DomainParticipant.

- DDS_EntityFactoryQosPolicy entity_factory

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

- CoreDX_PeerParticipantQosPolicy peer_participants

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

- CoreDX_DiscoveryQosPolicy discovery

  Override QoS values for builtin discovery entities.

- CoreDX_ThreadModelQosPolicy thread_model

  Configure DomainParticipant thread usage.

3.19.1 Detailed Description

Structure that holds DDS_DomainParticipant Quality of Service policies.

See also

- DDS_DomainParticipant_set_qos()
- DDS_DomainParticipant_get_qos()
- DDS_DomainParticipantFactory_create_participant()
- DDS_DomainParticipantFactory_set_default_participant_qos()
- DDS_DomainParticipantFactory_get_default_participant_qos()

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

- dds.h

3.20 DDS_DynamicType Struct Reference

DDS_DynamicType is an object that enhances CoreDX DDS with the facilities to process dynamic data types (in other words, data types that are not defined at compile time).

Related Functions

(Note that these are not member functions.)

- DDS_TypeCodeKind DDS_DynamicType_get_type (DDS_DynamicType t)
  Provides access to the 'type' of the DynamicType object. Applicable to any DynamicType.
- unsigned char DDS_DynamicType_get_octet (DDS_DynamicType t)
  Provides access to data held in an OCTET DynamicType object.
- unsigned char DDS_DynamicType_get_boolean (DDS_DynamicType t)
  Provides access to data held in an BOOLEAN DynamicType object.
- char DDS_DynamicType_get_char (DDS_DynamicType t)
  Provides access to data held in an CHAR DynamicType object.
- int16_t DDS_DynamicType_get_short (DDS_DynamicType t)
  Provides access to data held in an SHORT DynamicType object.
- uint16_t DDS_DynamicType_get_ushort (DDS_DynamicType t)
  Provides access to data held in an UNSIGNED SHORT DynamicType object.
- int32_t DDS_DynamicType_get_long (DDS_DynamicType t)
  Provides access to data held in an LONG DynamicType object.
- uint32_t DDS_DynamicType_get_ulong (DDS_DynamicType t)
  Provides access to data held in an UNSIGNED LONG DynamicType object.
- int64_t DDS_DynamicType_get_longlong (DDS_DynamicType t)
  Provides access to data held in an LONG LONG DynamicType object.
- uint64_t DDS_DynamicType_get_ulonglong (DDS_DynamicType t)
  Provides access to data held in an UNSIGNED LONG LONG DynamicType object.
- float DDS_DynamicType_get_float (DDS_DynamicType t)
  Provides access to data held in an FLOAT DynamicType object.

- double DDS_DynamicType_get_double (DDS_DynamicType t)
  Provides access to data held in an DOUBLE DynamicType object.
- const char * DDS_DynamicType_get_string (DDS_DynamicType t)
  Provides access to data held in an LONG DOUBLE DynamicType object.
- uint32_t DDS_DynamicType_get_max_length (DDS_DynamicType t)
  Provides access to the maximum length of a DynamicType object.
- uint32_t DDS_DynamicType_get_length (DDS_DynamicType t)
  Provides access to the length of data held in a DynamicType object.
- DDS_DynamicType DDS_DynamicType_get_element_type (DDS_DynamicType t)
  Provides access to the type of the of data held in an ARRAY or SEQUENCE DynamicType object.
- DDS_DynamicType DDS_DynamicType_get_element (DDS_DynamicType t, uint32_t n)
  Provides access to the a data element held in an ARRAY or SEQUENCE DynamicType object.
- uint32_t DDS_DynamicType_get_num_fields (DDS_DynamicType t)
  Provides access to the number of fields held by a STRUCT or UNION DynamicType object.
- DDS_DynamicType DDS_DynamicType_get_field (DDS_DynamicType t, uint32_t n)
  Provides access to a field held by a STRUCT or UNION DynamicType object.
- const char * DDS_DynamicType_get_field_name (DDS_DynamicType t, uint32_t n)
  Provides access to the name of a field held by a STRUCT or UNION DynamicType object.
- unsigned char DDS_DynamicType_get_field_key (DDS_DynamicType t, uint32_t n)
  Provides access to the 'key' indication for a field held by a STRUCT DynamicType object.
- DDS_DynamicType DDS_DynamicType_get_discriminator (DDS_DynamicType t)
  Provides access to the 'discriminator' type of a UNION DynamicType object.
- int32_t DDS_DynamicType_get_default_field (DDS_DynamicType t)
  Provides access to the 'default' field of a UNION DynamicType object.
- uint32_t DDS_DynamicType_get_field_num_labels (DDS_DynamicType t, uint32_t field)
  Provides access to the number of labels assigned to a field in the UNION DynamicType object.
- int32_t DDS_DynamicType_get_field_label (DDS_DynamicType t, uint32_t field, uint32_t label)
  Provides access to the number of labels assigned to a field in the UNION DynamicType object.

- DDS_DynamicType DDS_DynamicType_get_selected_field (DDS_DynamicType t)

  Provides access to the selected field of a UNION DynamicType object.

- DDS_TypeDefinition DDS_create_type_definition (const unsigned char *typecodes, uint32_t tc_len, unsigned char tc_encoding)

  Create a TypeDefinition structure based on the provided TypeCode data.

- DDS_TypeDefinition DDS_create_type_definition_from_typecode (const unsigned char *typecodes, uint32_t tc_len, unsigned char tc_encoding)

  Create a TypeDefinition structure based on the provided TypeCode data.

- DDS_TypeDefinition DDS_create_type_definition_from_dynamictype (DDS_DynamicType dtype)

  Create a TypeDefinition structure based on the provided DynamicType object.

- DDS_TypeSupport DDS_create_dynamic_typesupport (DDS_TypeDefinition type_def)

  Create a TypeSupport object based on the provided TypeDefinition object.

- DDS_TypeSupport DDS_DynamicType_create_typesupport (DDS_DynamicType t)

  Create a TypeSupport object based on the provided DynamicType type.

- void DDS_destroy_dynamic_typesupport (DDS_TypeSupport ts)

  Destroys a TypeSupport object created by a call to DDS_create_dynamic_type_typesupport().

- DDS_ReturnCode_t DynamicTypeTypeSupport_register_type (DDS_DomainParticipant domain, const char *type_name, DDS_TypeSupport type_support)

  Registers a TypeSupport object (created by a call to DDS_DynamicType_create_typesupport()), with the CoreDX DDS middleware.

- DDS_ReturnCode_t DDS_DynamicTypeTypeSupport_register_type (DDS_TypeSupport type_support, DDS_DomainParticipant domain, const char *type_name)

  Registers a TypeSupport object (created by a call to DDS_DynamicType_create_typesupport()), with the CoreDX DDS middleware.

- DDS_DynamicType DDS_TypeDefinition_create_dynamictype (DDS_TypeDefinition type_def)

  Create a DDS_DynamicType object from a DDS_TypeDefinition.

- DDS_DynamicType DDS_DynamicType_alloc (DDS_TypeCodeKind type_code)

  Allocates and returns a DynamicType configured to hold the specified 'type_code' type.

- DDS_DynamicType DDS_DynamicType_alloc_basic (DDS_TypeCodeKind type_code)

  Allocates and returns a DynamicType configured to hold the specified 'type_code' basic type.

- DDS_DynamicType DDS_DynamicType_alloc_string ()

Allocates and returns a STRING DynamicType.

- DDS_DynamicType DDS_DynamicType_alloc_array ()
  Allocates and returns an ARRAY DynamicType.
- DDS_DynamicType DDS_DynamicType_alloc_sequence ()
  Allocates and returns a SEQUENCE DynamicType.
- DDS_DynamicType DDS_DynamicType_alloc_struct ()
  Allocates and returns a STRUCT DynamicType.
- DDS_DynamicType DDS_DynamicType_alloc_union ()
  Allocates and returns a UNION DynamicType.
- void DDS_DynamicType_free (DDS_DynamicType t)
  Reclaims all memory used by an allocated DynamicType object.
- DDS_ReturnCode_t DDS_DynamicType_set_octet (DDS_DynamicType t, unsigned char c)
  Assigns a value to the provided OCTET DynamicType.
- DDS_ReturnCode_t DDS_DynamicType_set_boolean (DDS_DynamicType t, unsigned char c)
  Assigns a value to the provided BOOLEAN DynamicType.
- DDS_ReturnCode_t DDS_DynamicType_set_char (DDS_DynamicType t, char c)
  Assigns a value to the provided CHAR DynamicType.
- DDS_ReturnCode_t DDS_DynamicType_set_short (DDS_DynamicType t, short c)
  Assigns a value to the provided SHORT DynamicType.
- DDS_ReturnCode_t DDS_DynamicType_set_ushort (DDS_DynamicType t, unsigned short c)
  Assigns a value to the provided USHORT DynamicType.
- DDS_ReturnCode_t DDS_DynamicType_set_long (DDS_DynamicType t, long c)
  Assigns a value to the provided LONG DynamicType.
- DDS_ReturnCode_t DDS_DynamicType_set_ulong (DDS_DynamicType t, unsigned long c)
  Assigns a value to the provided ULONG DynamicType.
- DDS_ReturnCode_t DDS_DynamicType_set_longlong (DDS_DynamicType t, int64_t c)
  Assigns a value to the provided LONGLONG DynamicType.
- DDS_ReturnCode_t DDS_DynamicType_set_ulonglong (DDS_DynamicType t, uint64_t c)

Assigns a value to the provided ULONGLONG DynamicType.

- DDS_ReturnCode_t DDS_DynamicType_set_float (DDS_DynamicType t, float c)

  Assigns a value to the provided FLOAT DynamicType.

- DDS_ReturnCode_t DDS_DynamicType_set_double (DDS_DynamicType t, double c)

  Assigns a value to the provided DOUBLE DynamicType.

- DDS_ReturnCode_t DDS_DynamicType_set_string (DDS_DynamicType t, const char *c)

  Assigns a value to the provided LONGDOUBLE DynamicType.

- DDS_ReturnCode_t DDS_DynamicType_set_max_length (DDS_DynamicType t, uint32_t n)

  Assigns a 'max_length' value to the provided STRING, ARRAY, or SEQUENCE DynamicType.

- DDS_ReturnCode_t DDS_DynamicType_set_length (DDS_DynamicType t, uint32_t n)

  Assigns a 'length' value to the provided ARRAY or SEQUENCE DynamicType.

- DDS_ReturnCode_t DDS_DynamicType_set_element_type (DDS_DynamicType t, DDS_DynamicType e)

  Defines the element type of an ARRAY or SEQUENCE.

- DDS_ReturnCode_t DDS_DynamicType_set_element (DDS_DynamicType t, uint32_t n, DDS_DynamicType e)

  Assigns a value to an element of an ARRAY or SEQUENCE.

- DDS_ReturnCode_t DDS_DynamicType_set_num_fields (DDS_DynamicType t, uint32_t n)

  Defines the number of fields held by a STRUCT or UNION DynamicType object.

- DDS_ReturnCode_t DDS_DynamicType_set_field (DDS_DynamicType t, uint32_t n, const char *field_name, DDS_DynamicType e, unsigned char key)

  Assigns a value to a field of a STRUCT or UNION DynamicType object.

- DDS_ReturnCode_t DDS_DynamicType_set_discriminator (DDS_DynamicType t, DDS_DynamicType d)

  Defines the type and value of the UNION discriminator.

- DDS_ReturnCode_t DDS_DynamicType_set_default_field (DDS_DynamicType t, int field)

  Defines the index of the default field within the UNION.

- DDS_ReturnCode_t DDS_DynamicType_set_field_num_labels (DDS_DynamicType t, uint32_t field, uint32_t n)

  Defines the number of labels associated with a field within the UNION.

- DDS_ReturnCode_t DDS_DynamicType_set_field_label (DDS_DynamicType t, uint32_t field, uint32_t label, int32_t val)

  Assigns a label to the specified field within the UNION.

3.20.1 Detailed Description

DDS_DynamicType is an object that enhances CoreDX DDS with the facilities to process dynamic data types (in other words, data types that are not defined at compile time). The ability to process dynamic types adds significant flexibility to CoreDX DDS. The type information can be discovered at run-time, constructed programmatically, or based on existing type information. The DynamicType support allows the application to process a data type without compiling and linking type specific source code. For an application that must support a large number of data types, this can offer a savings in code size.

3.20.2 Friends And Related Function Documentation

3.20.2.1 DDS_TypeSupport DDS_create_dynamic_typesupport (DDS_TypeDefinition *type_def*) [related]

Create a TypeSupport object based on the provided TypeDefinition object.

Construct a TypeSupport object that can be used to register a Data Type with the CoreDX DDS middleware.

3.20.2.2 DDS_TypeDefinition DDS_create_type_definition (const unsigned char * *typecodes,* uint32_t *tc_len,* unsigned char *tc_encoding*) [related]

Create a TypeDefinition structure based on the provided TypeCode data.

This routine should not be used. Instead, use DDS_create_type_definition_from_typecode().

3.20.2.3 DDS_TypeDefinition DDS_create_type_definition_from_dynamictype (DDS_DynamicType *dtype*) [related]

Create a TypeDefinition structure based on the provided DynamicType object.

Construct a TypeDefinition object that can be used to construct a TypeSupport object. The resulting TypeDefintion will support the data type represented by the provided DynamicType object. The TypeDefinition can be used in a call to DDS_create_dynamic_typesupport().

3.20.2.4 DDS_TypeDefinition DDS_create_type_definition_from_typecode (const unsigned char * *typecodes*, uint32_t *tc_len*, unsigned char *tc_encoding*) [related]

Create a TypeDefinition structure based on the provided TypeCode data.

Construct a TypeDefinition object that can be used to construct a TypeSupport object. The resulting TypeDefintion will support the data type represented by the provided TypeCode information. The TypeDefinition can be used in a call to DDS_create_dynamic_typesupport().

3.20.2.5 DDS_DynamicType DDS_DynamicType_alloc (DDS_TypeCodeKind *type_code*) [related]

Allocates and returns a DynamicType configured to hold the specified 'type_code' type.

This can be used to create a DynamicType for any of the the defined data types.

3.20.2.6 DDS_DynamicType DDS_DynamicType_alloc_array ( ) [related]

Allocates and returns an ARRAY DynamicType.

The type of the array elements must be specified by calling DDS_DynamicType_set_element_type(). The size of the array must be specified by calling DDS_DynamicType_set_max_length(). Before adding data to the array, memory for the elements must be allocated by calling DDS_DynamicType_set_length(). Data can be added to the array by calling DDS_DynamicType_set_element().

3.20.2.7 DDS_DynamicType DDS_DynamicType_alloc_basic (DDS_TypeCodeKind *type_code*) [related]

Allocates and returns a DynamicType configured to hold the specified 'type_code' basic type.

This can be used to create a DynamicType for the following data types: SHORT, LONG, LONGLONG, USHORT, ULONG, ULONGLONG, FLOAT, DOUBLE, BOOLEAN, CHAR, OCTET, or ENUM.

3.20.2.8 DDS_DynamicType DDS_DynamicType_alloc_sequence ( ) [related]

Allocates and returns a SEQUENCE DynamicType.

The type of the sequence elements must be specified by calling DDS_DynamicType_set_element_type(). The size of the sequence must be specified by by calling DDS_DynamicType_set_max_length(). Before adding data to the sequence, memory for the elements must be allocated by calling DDS_DynamicType_set_length(). Data can be added to the sequence by calling DDS_DynamicType_set_element().

3.20.2.9 DDS_DynamicType DDS_DynamicType_alloc_string ( ) [related]

Allocates and returns a STRING DynamicType.

By default the string is unbounded. The length of the string may be bounded (if desired) by calling DDS_DynamicType_set_max_length().

3.20.2.10 DDS_DynamicType DDS_DynamicType_alloc_struct ( ) [related]

Allocates and returns a STRUCT DynamicType.

The DDS_DynamicType_set_num_fields() must be called to define the number of fields in the structure. For each field, DDS_DynamicType_set_field() must be used to define the type of the field.

3.20.2.11 DDS_DynamicType DDS_DynamicType_alloc_union ( ) [related]

Allocates and returns a UNION DynamicType.

The DDS_Dynamic_type_set_discriminator() function must be used to define the discriminator that is ued to select one of the union fields. The DDS_DynamicType_set_num_fields() must be called to define the number of fields in the union. For each field:

1. DDS_DynamicType_set_field() must be used to define the type of the field.
2. The DDS_DynamicType_field_num_labels() function must be used to define the number of case labels (discriminator values) that select this field. (For a 'default:' only field, the number can be zero.)
3. The DDS_DynamicType_set_field_label() function must be used to define a specific case label for the field.

The DDS_DynamicType_set_default_field() function is used to (optionally) specify a field that is used as a 'default:' case.

3.20.2.12 DDS_TypeSupport DDS_DynamicType_create_typesupport (DDS_DynamicType *t*) [related]

Create a TypeSupport object based on the provided DynamicType type.

Construct a TypeSupport object that can be used to register a Data Type with the CoreDX DDS middleware.

3.20.2.13 void DDS_DynamicType_free (DDS_DynamicType *t*) [related]

Reclaims all memory used by an allocated DynamicType object.

Call this routine to free memory associated with a DynamicType object created by one of the DDS_DynamicType_alloc_xxx() routines or by DDS_TypeDefinition_create_dynamictype(). This routines frees the type information as well as any data values contained in the DynamicType object.

3.20.2.14 unsigned char DDS_DynamicType_get_boolean (DDS_DynamicType *t*) [related]

Provides access to data held in an BOOLEAN DynamicType object.

Return values

unsigned_char 0 indicates FALSE, non-zero indicates TRUE.

3.20.2.15 char DDS_DynamicType_get_char (DDS_DynamicType *t*) [related]

Provides access to data held in an CHAR DynamicType object.

Return values

char the data value held by 't'.

3.20.2.16 int32_t DDS_DynamicType_get_default_field (DDS_DynamicType *t*) [related]

Provides access to the 'default' field of a UNION DynamicType object.

This is applicable only for a UNION DynamicType object. This returns the index of the 'default' field in the UNION, if there is no field marked with a 'default:' case label.

Return values

int32_t the index of the 'default' field. -1 if there is no defined 'default' case.

3.20.2.17 DDS_DynamicType DDS_DynamicType_get_discriminator (DDS_DynamicType *t*) [related]

Provides access to the 'discriminator' type of a UNION DynamicType object.

This is applicable only for a UNION DynamicType object. This returns the DynamicType that serves as a discriminator for the UNION data structure.

Return values

DDS_DynamicType the discriminator data.

3.20.2.18 double DDS_DynamicType_get_double (DDS_DynamicType *t*) [related]

Provides access to data held in an DOUBLE DynamicType object.

Return values

double the data value held by 't'.

3.20.2.19 DDS_DynamicType DDS_DynamicType_get_element (DDS_DynamicType *t,* uint32_t *n*) [related]

Provides access to the a data element held in an ARRAY or SEQUENCE DynamicType object.

This is applicable for a SEQUENCE or ARRAY DynamicType object. For a SEQUENCE, this returns the element 'n' of the sequence. For an ARRAY, this returns the element 'n' of the array elements. Elements are indexed starting at 0.

Return values

uint32_t the data element held in the SEQUENCE or ARRAY 't' at index 'n'.

3.20.2.20 DDS_DynamicType DDS_DynamicType_get_element_type (DDS_DynamicType *t*) [related]

Provides access to the type of the of data held in an ARRAY or SEQUENCE DynamicType object.

This is applicable for a SEQUENCE or ARRAY DynamicType object. For a SEQUENCE, this returns the 'type' of the sequence elements. For an ARRAY, this returns the 'type' of the array elements.

Return values

uint32_t the 'type' of the data value held in the SEQUENCE or ARRAY 't'.

3.20.2.21 DDS_DynamicType DDS_DynamicType_get_field (DDS_DynamicType *t,* uint32_t *n*) [related]

Provides access to a field held by a STRUCT or UNION DynamicType object.

This is applicable for a STRUCT or UNION DynamicType object. For a STRUCT or UNION, this returns a field held in the data structure.

Return values

DDS_DynamicType the number of data fields in the STRUCT or UNION 't'.

3.20.2.22 unsigned char DDS_DynamicType_get_field_key (DDS_DynamicType *t,* uint32_t *n*) [related]

Provides access to the 'key' indication for a field held by a STRUCT DynamicType object.

This is applicable for a STRUCT DynamicType object. For a STRUCT, this returns an indication that field 'n' is (or is not) a key in the data structure.

Return values

unsigned_char if non-zero, the field at index 'n' is a key field. if zero, the field at index 'n' is not a key field.

3.20.2.23 int32_t DDS_DynamicType_get_field_label (DDS_DynamicType *t,* uint32_t *field,* uint32_t *label*) [related]

Provides access to the number of labels assigned to a field in the UNION DynamicType object.

This is applicable only for a UNION DynamicType object. This returns the count of the number of labels used to select a particular field in the UNION. The 'default:' label is not included in this count - as a result, it is possible for one field to have 'zero' labels.

Return values

int32_t the value of the requested label of field identified by index 'field'.

3.20.2.24 const char * DDS_DynamicType_get_field_name (DDS_DynamicType *t,* uint32_t *n*) [related]

Provides access to the name of a field held by a STRUCT or UNION DynamicType object.

This is applicable for a STRUCT or UNION DynamicType object. For a STRUCT or UNION, this returns the name of a field held in the data structure.

Return values

const_char_* field name of field 'n' in STRUCT or UNION 't'.

3.20.2.25 uint32_t DDS_DynamicType_get_field_num_labels (DDS_DynamicType *t,* uint32_t *field*) [related]

Provides access to the number of labels assigned to a field in the UNION DynamicType object.

This is applicable only for a UNION DynamicType object. This returns the count of the number of labels used to select a particular field in the UNION. The 'default:' label is not included in this count.

Return values

int32_t the number of labels assigned to field identified by index 'field'.

3.20.2.26 float DDS_DynamicType_get_float (DDS_DynamicType *t*) [related]

Provides access to data held in an FLOAT DynamicType object.

Return values

float the data value held by 't'.

3.20.2.27 uint32_t DDS_DynamicType_get_length (DDS_DynamicType *t*) [related]

Provides access to the length of data held in a DynamicType object.

This is applicable for a SEQUENCE or ARRAY DynamicType object. For a SEQUENCE, this returns the 'length' of the sequence. For an ARRAY, this returns the size of the array.

Return values

uint32_t the length of the data value held by 't'.

3.20.2.28 int32_t DDS_DynamicType_get_long (DDS_DynamicType *t*) [related]

Provides access to data held in an LONG DynamicType object.

Return values

int32_t the data value held by 't'.

3.20.2.29 int64_t DDS_DynamicType_get_longlong (DDS_DynamicType *t*) [related]

Provides access to data held in an LONG LONG DynamicType object.

Return values

int64_t the data value held by 't'.

3.20.2.30 uint32_t DDS_DynamicType_get_max_length (DDS_DynamicType *t*) [related]

Provides access to the maximum length of a DynamicType object.

This is applicable for a STRING, SEQUENCE, or ARRAY DynamicType object. For a STRING, this returns the 'fixed length' of the string, or zero if the string is not fixed length. For a SEQUENCE, this returns the 'fixed length' of the sequence, or zero if the sequence is unbounded. For an ARRAY, this returns the size of the array.

Return values

int32_t the maximum length of the data value held by 't'.

3.20.2.31 uint32_t DDS_DynamicType_get_num_fields (DDS_DynamicType *t*) [related]

Provides access to the number of fields held by a STRUCT or UNION DynamicType object.

This is applicable for a STRUCT or UNION DynamicType object. For a STRUCT, this returns the number of fields held in the structure. For a UNION, this returns the number of fields contained by the UNION where a field is selectable by one or more case labels.

Return values

uint32_t the number of data fields in the STRUCT or UNION 't'.

3.20.2.32 unsigned char DDS_DynamicType_get_octet (DDS_DynamicType *t*) [related]

Provides access to data held in an OCTET DynamicType object.

Return values

unsigned_char the data value held by 't'.

3.20.2.33 DDS_DynamicType DDS_DynamicType_get_selected_field (DDS_DynamicType *t*) [related]

Provides access to the selected field of a UNION DynamicType object.

This is useful to access the field that is selected by the current value of the union's 'discriminator'.

3.20.2.34 int16_t DDS_DynamicType_get_short (DDS_DynamicType *t*) [related]

Provides access to data held in an SHORT DynamicType object.

Return values

int16_t the data value held by 't'.

3.20.2.35 const char * DDS_DynamicType_get_string (DDS_DynamicType *t*) [related]

Provides access to data held in an LONG DOUBLE DynamicType object.

Return values

long double the data value held by 't'.

Not Yet Supported

Long Double data types are not supported by CoreDX DDS.

Provides access to data held in an STRING DynamicType object.

Return values

const_char_* the data value held by 't'.

3.20.2.36 DDS_TypeCodeKind DDS_DynamicType_get_type (DDS_DynamicType *t*) [related]

Provides access to the 'type' of the DynamicType object. Applicable to any DynamicType.

Return values

DDS_TypeCodeKind the type of object 't'.

3.20.2.37 uint32_t DDS_DynamicType_get_ulong (DDS_DynamicType *t*) [related]

Provides access to data held in an UNSIGNED LONG DynamicType object.

Return values

uint32_t the data value held by 't'.

3.20.2.38 uint64_t DDS_DynamicType_get_ulonglong (DDS_DynamicType *t*) [related]

Provides access to data held in an UNSIGNED LONG LONG DynamicType object.

Return values

uint64_t the data value held by 't'.

3.20.2.39 uint16_t DDS_DynamicType_get_ushort (DDS_DynamicType *t*) [related]

Provides access to data held in an UNSIGNED SHORT DynamicType object.

Return values

uint16_t the data value held by 't'.

3.20.2.40 DDS_ReturnCode_t DDS_DynamicType_set_boolean (DDS_DynamicType *t,* unsigned char *c*) [related]

Assigns a value to the provided BOOLEAN DynamicType.

Return values

BAD_PARAMETER if 't' is not of type BOOLEAN.

OK upon success.

3.20.2.41 DDS_ReturnCode_t DDS_DynamicType_set_char (DDS_DynamicType *t,* char *c*) [related]

Assigns a value to the provided CHAR DynamicType.

Return values

BAD_PARAMETER if 't' is not of type CHAR.

OK upon success.

3.20.2.42 DDS_ReturnCode_t DDS_DynamicType_set_default_field (DDS_DynamicType *t,* int *field*) [related]

Defines the index of the default field within the UNION.

The active field in the UNION is selected by considering the value of the discriminator and the values of the field labels. Any value of the discriminator not explicitly listed in the labels will select the default field. The default field may have other labels assigned to it, or it may have zero labels.

Note

An index value of -1 indicates that the UNION has no default field.

Return values

BAD_PARAMETER if 't' is not of type UNION.

OK upon success.

3.20.2.43 DDS_ReturnCode_t DDS_DynamicType_set_discriminator (DDS_DynamicType *t,* DDS_DynamicType *d*) [related]

Defines the type and value of the UNION discriminator.

The discriminator is used to identify which one of the UNION fields is active. Only one field within the UNION is active at a given time. The active field is selected by considering the value of the discriminator and the values of the field labels. Each field can have a set of labels containing zero or more unique discriminator values. Field labels are defined with the DDS_DynamicType_set_field_num_labels() and DDS_DynamicType_set_field_label() functions.

Note

The UNION takes ownership of the provided discriminator.
One field in the union can be designated as the 'default' field. Any value of the discriminator not explicitly listed in the labels will select the default field. The default field may have other labels assigned to it, or it may have zero labels.

Return values

BAD_PARAMETER if 't' is not of type UNION.

OK upon success.

3.20.2.44 DDS_ReturnCode_t DDS_DynamicType_set_double (DDS_DynamicType *t*, double *c*) [related]

Assigns a value to the provided DOUBLE DynamicType.

Return values

BAD_PARAMETER if 't' is not of type DOUBLE.

OK upon success.

3.20.2.45 DDS_ReturnCode_t DDS_DynamicType_set_element (DDS_DynamicType *t*, uint32_t *n*, DDS_DynamicType *e*) [related]

Assigns a value to an element of an ARRAY or SEQUENCE.

Note

The ARRAY or SEQUENCE takes ownership of the provided field.

Return values

BAD_PARAMETER if 't' is not of type ARRAY or SEQUENCE or if 'n' is beyond the specified length of the ARRAY or SEQUENCE.

OK upon success.

3.20.2.46 DDS_ReturnCode_t DDS_DynamicType_set_element_type (DDS_DynamicType *t,* DDS_DynamicType *e*) [related]

Defines the element type of an ARRAY or SEQUENCE.

Return values

BAD_PARAMETER if 't' is not of type ARRAY or SEQUENCE.

OK upon success.

3.20.2.47 DDS_ReturnCode_t DDS_DynamicType_set_field (DDS_DynamicType *t,* uint32_t *n,* const char * *field_name,* DDS_DynamicType *e,* unsigned char *key*) [related]

Assigns a value to a field of a STRUCT or UNION DynamicType object.

The 'nth' field of the struct or union is assigned the provided **field_name**, type **e**. The **key** parameter is used only for STRUCT types. If key is non-zero, then the field is added to the 'key set' for this data type. This routine makes a copy of the 'field_name' argument.

Note

The STRUCT or UNION takes ownership of the provided field.
Fields of a UNION data type can not be part of a 'key set', as the field may not exist in a particular instantiation of the union data type.

Return values

BAD_PARAMETER if 't' is not of type STRUCT or UNION.

OUT_OF_MEMORY if memory allocation (to hold a copy of field_name) fails.

OK upon success.

3.20.2.48 DDS_ReturnCode_t DDS_DynamicType_set_field_label (DDS_DynamicType *t,* uint32_t *field,* uint32_t *label,* int32_t *val*) [related]

Assigns a label to the specified field within the UNION.

Return values

BAD_PARAMETER if 't' is not of type UNION or if **field** does not specify a valid field index, or if **label** does not specify a valid label index for the field.

OUT_OF_RESOURCES if memory allocation fails.

OK upon success.

3.20.2.49 DDS_ReturnCode_t DDS_DynamicType_set_field_num_labels (DDS_DynamicType *t*, uint32_t *field*, uint32_t *n*) [related]

Defines the number of labels associated with a field within the UNION.

Each field in a UNION has zero or more labels associated. The active field in the UNION is selected by considering the value of the discriminator and the values of the field labels. A field is selected when the value of the discriminator matches the value of one of its labels. Any value of the discriminator not explicitly listed in the all of the labels will select the default field. [The default field may have other labels assigned to it, or it may have zero labels.]

Note

Subsequent calls to this routine for the same field will clear any previously assigned labels. [The default field may have other labels assigned to it, or it may have zero labels.]

Return values

BAD_PARAMETER if 't' is not of type UNION.

OUT_OF_RESOURCES if memory allocation fails.

OK upon success.

3.20.2.50 DDS_ReturnCode_t DDS_DynamicType_set_float (DDS_DynamicType *t*, float *c*) [related]

Assigns a value to the provided FLOAT DynamicType.

Return values

BAD_PARAMETER if 't' is not of type FLOAT.

OK upon success.

3.20.2.51 DDS_ReturnCode_t DDS_DynamicType_set_length (DDS_DynamicType *t*, uint32_t *n*) [related]

Assigns a 'length' value to the provided ARRAY or SEQUENCE DynamicType.

This defines the actual size of the array or sequence data. This will allocate memory to hold 'n' entries. The entries are not initialized, and must be initialized by calling DDS_DynamicType_set_element() 'n' times.

Note

If DDS_DynamicType_set_length() has been called previously on object 't', the subsequent calls to this routine will deallocate the storate for the previous array or sequence elements. However, the elements themselves will not be freed. To avoid a memory leak, it is necessary to manually free each element before calling set_length().

Return values

BAD_PARAMETER if 't' is not of type ARRAY or SEQUENCE; OUT_OF_RESOURCES if memory allocation fails.

OK upon success.

3.20.2.52 DDS_ReturnCode_t DDS_DynamicType_set_long (DDS_DynamicType *t*, long *c*) [related]

Assigns a value to the provided LONG DynamicType.

Return values

BAD_PARAMETER if 't' is not of type LONG.

OK upon success.

3.20.2.53 DDS_ReturnCode_t DDS_DynamicType_set_longlong (DDS_DynamicType *t*, int64_t *c*) [related]

Assigns a value to the provided LONGLONG DynamicType.

Return values

BAD_PARAMETER if 't' is not of type LONGLONG.

OK upon success.

3.20.2.54 DDS_ReturnCode_t DDS_DynamicType_set_max_length (DDS_DynamicType *t*, uint32_t *n*) [related]

Assigns a 'max_length' value to the provided STRING, ARRAY, or SEQUENCE DynamicType.

For STRINGS and SEQUENCES, this defines the bound on the string length or sequence length. For arrays, this defines the size of the array.

Return values

BAD_PARAMETER if 't' is not of type STRING, ARRAY, or SEQUENCE.

OK upon success.

3.20.2.55 DDS_ReturnCode_t DDS_DynamicType_set_num_fields (DDS_DynamicType *t*, uint32_t *n*) [related]

Defines the number of fields held by a STRUCT or UNION DynamicType object.

Return values

BAD_PARAMETER if 't' is not of type STRUCT or UNION.

OK upon success.

3.20.2.56 DDS_ReturnCode_t DDS_DynamicType_set_octet (DDS_DynamicType *t*, unsigned char *c*) [related]

Assigns a value to the provided OCTET DynamicType.

Return values

BAD_PARAMETER if 't' is not of type OCTET.

OK upon success.

3.20.2.57 DDS_ReturnCode_t DDS_DynamicType_set_short (DDS_DynamicType *t*, short *c*) [related]

Assigns a value to the provided SHORT DynamicType.

Return values

BAD_PARAMETER if 't' is not of type SHORT.

OK upon success.

3.20.2.58 DDS_ReturnCode_t DDS_DynamicType_set_string (DDS_DynamicType *t*, const char * *c*) [related]

Assigns a value to the provided LONGDOUBLE DynamicType.

Return values

BAD_PARAMETER if 't' is not of type LONGDOUBLE.

OK upon success.

Not Yet Supported

Long Double data types are not supported by CoreDX DDS.

Assigns a value to the provided STRING DynamicType.

This routine will make a copy of the provided string data. If the STRING has a defined 'max_length', then the copied data will be truncated to 'max_length' characters. [A 0x00 (nul) character will be added after the truncated data.]

Return values

DDS_ReturnCode_t BAD_PARAMETER if 't' is not of type STRING.

OUT_OF_RESOURCES if memory allocation fails.

OK upon success.

3.20.2.59 DDS_ReturnCode_t DDS_DynamicType_set_ulong (DDS_DynamicType *t,* unsigned long *c*) [related]

Assigns a value to the provided ULONG DynamicType.

Return values

BAD_PARAMETER if 't' is not of type ULONG.

OK upon success.

3.20.2.60 DDS_ReturnCode_t DDS_DynamicType_set_ulonglong (DDS_DynamicType *t,* uint64_t *c*) [related]

Assigns a value to the provided ULONGLONG DynamicType.

Return values

BAD_PARAMETER if 't' is not of type ULONGLONG.

OK upon success.

3.20.2.61 DDS_ReturnCode_t DDS_DynamicType_set_ushort (DDS_DynamicType *t,* unsigned short *c*) [related]

Assigns a value to the provided USHORT DynamicType.

Return values

BAD_PARAMETER if 't' is not of type USHORT.

OK upon success.

3.20.2.62 DDS_ReturnCode_t DDS_DynamicTypeTypeSupport_register_type (DDS_TypeSupport *type_support*, DDS_DomainParticipant *domain*, const char * *type_name*) [related]

Registers a TypeSupport object (created by a call to DDS_DynamicType_create_typesupport()), with the CoreDX DDS middleware.

After registering the TypeSupport object, the data type will be available to the middleware for use by DataReaders and DataWriters.

3.20.2.63 DDS_DynamicType DDS_TypeDefinition_create_dynamictype (DDS_TypeDefinition *type_def*) [related]

Create a DDS_DynamicType object from a DDS_TypeDefinition.

Creates a DDS_DynamicType object initialized to represent the Data Type identified by the provided DDS_-TypeDefinition.

3.20.2.64 DDS_ReturnCode_t DynamicTypeTypeSupport_register_type (DDS_DomainParticipant *domain*, const char * *type_name*, DDS_TypeSupport *type_support*) [related]

Registers a TypeSupport object (created by a call to DDS_DynamicType_create_typesupport()), with the CoreDX DDS middleware.

After registering the TypeSupport object, the data type will be available to the middleware for use by DataReaders and DataWriters.

Note

This routine is deprecated - Use DDS_DynamicTypeTypeSupport_register_type() instead.

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

- dds_dtype.h

3.21 DDS_DynamicTypeDataReader Struct Reference

Provides a DataReader interface that is tailored to support reading a DynamicType data type. The specific DynamicType must have been registered previously with the DomainParticipant.

3.21.1 Detailed Description

Provides a DataReader interface that is tailored to support reading a DynamicType data type. The specific DynamicType must have been registered previously with the DomainParticipant.

See also

DDS_DataReader

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

- dds_dtype.h

3.22 DDS_DynamicTypeDataWriter Struct Reference

Provides a DataWriter interface that is tailored to support writing a DynamicType data type. The specific DynamicType must have been registered previously with the DomainParticipant.

3.22.1 Detailed Description

Provides a DataWriter interface that is tailored to support writing a DynamicType data type. The specific DynamicType must have been registered previously with the DomainParticipant.

See also

DDS_DataWriter

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

- dds_dtype.h

3.23 DDS_GuardCondition Struct Reference

A DDS_GuardCondition is a condition where the **trigger_value** is under application control.

Related Functions

(Note that these are not member functions.)

- DDS_GuardCondition DDS_GuardCondition__alloc ()

  This routine allocates a DDS_GuardCondition.

- void DDS_GuardCondition__free (struct _GuardCondition *gc)

  This routine destroys the provided DDS_GuardCondition.

- unsigned char DDS_GuardCondition_get_trigger_value (DDS_GuardCondition gc)

  *This routine returns the current value of the **trigger_value** in **gc**.*

- DDS_ReturnCode_t DDS_GuardCondition_set_trigger_value (DDS_GuardCondition gc, unsigned char v)

  *This routine set the current value of the **trigger_value** in **gc**.*

3.23.1 Detailed Description

A DDS_GuardCondition is a condition where the **trigger_value** is under application control.

3.23.2 Friends And Related Function Documentation

3.23.2.1 DDS_GuardCondition DDS_GuardCondition__alloc ( ) [related]

This routine allocates a DDS_GuardCondition.

The returned GuardCondition should be destroyed by a call to DDS_GuardCondition__free().

3.23.2.2 void DDS_GuardCondition__free (struct _GuardCondition * *gc*) [related]

This routine destroys the provided DDS_GuardCondition.

The provided GuardCondition must have been previously allocated by a call to DDS_GuardCondition__alloc(). The caller must ensure that the GuardCondition is not attached to any WaitSets prior to calling DDS_GuardCondition__free().

3.23.2.3 unsigned char DDS_GuardCondition_get_trigger_value (DDS_GuardCondition *gc*) [related]

This routine returns the current value of the **trigger_value** in **gc**.

A non-zero return value indicates that the **trigger_value** is TRUE.

A zero return value indicates that the **trigger_value** is FALSE.

3.23.2.4 DDS_ReturnCode_t DDS_GuardCondition_set_trigger_value (DDS_GuardCondition *gc*, unsigned char *v*) [related]

This routine set the current value of the **trigger_value** in **gc**.

A non-zero **v** argument indicates that the **trigger_value** is TRUE.

A zero **v** argument indicates that the **trigger_value** is FALSE. Setting the trigger value to a non-zero state will cause any threads that are waiting on a WaitSet with this GuardCondition attached to unblock. [In other words, the DDS_WaitSet_wait() routine will return if the WaitSet has this GuardCondition attached to it.]

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

- dds.h

3.24 DDS_InconsistentTopicStatus Struct Reference

Status related to the on_inconsistent_topic listener methods of the DDS_TopicListener structure.

Public Attributes

- int total_count

  Cummulative count of the discovered Topics having a matching name and inconsistent characteristics.

- int total_count_change

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

3.24.1 Detailed Description

Status related to the on_inconsistent_topic listener methods of the DDS_TopicListener structure.

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

- dds.h

3.25 DDS_LivelinessChangedStatus Struct Reference

Status related to the on_liveliness_changed listener methods of the DDS_DataReader, DDS_Subscriber, and DDS_DomainParticipant structures.

Public Attributes

- int alive_count

  The number of 'active' DataWriters matched to this DataReader.

- int not_alive_count

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

- int alive_count_change

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

- int not_alive_count_change

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

- DDS_InstanceHandle_t last_publication_handle

  Handle identifying the most recent DataWriter whose liveliness changed.

3.25.1 Detailed Description

Status related to the on_liveliness_changed listener methods of the DDS_DataReader, DDS_Subscriber, and DDS_DomainParticipant structures.

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

- dds.h

3.26 DDS_LivelinessLostStatus Struct Reference

Status related to the on_liveliness_lost listener methods of the DDS_DataWriter, DDS_Publisher, and DDS_-DomainParticipant structures.

Public Attributes

- int total_count

  Cummulative number of times that an 'alive' DataWriter became not alive.

- int total_count_change

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

3.26.1 Detailed Description

Status related to the on_liveliness_lost listener methods of the DDS_DataWriter, DDS_Publisher, and DDS_-DomainParticipant structures.

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

- dds.h

3.27 DDS_MultiTopic Struct Reference

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

Related Functions

(Note that these are not member functions.)

- DDS_TopicDescription DDS_MultiTopic_TopicDescription (DDS_MultiTopic t)
  This operation 'casts' the provide DDS_MultiTopic to a DDS_TopicDescrition.

3.27.1 Detailed Description

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

Not Yet Supported

CoreDX does not yet implement MultiTopics.

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

- dds.h

3.28 DDS_OfferedDeadlineMissedStatus Struct Reference

Status related to the on_offered_deadline_missed listener methods of the DDS_DataWriter, DDS_Publisher, and DDS_DomainParticipant structures.

Public Attributes

- int total_count

  Cummulative count of the number of deadlines missed by this DataWriter.

- int total_count_change

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

- DDS_InstanceHandle_t last_instance_handle

  Handle identifying the most recent instance whose deadline was missed.

3.28.1 Detailed Description

Status related to the on_offered_deadline_missed listener methods of the DDS_DataWriter, DDS_Publisher, and DDS_DomainParticipant structures.

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

- dds.h

3.29 DDS_OfferedIncompatibleQosStatus Struct Reference

Status related to the on_offered_incompatible_qos listener methods of the DDS_DataWriter, DDS_Publisher, and DDS_DomainParticipant structures.

Public Attributes

- int total_count

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

- int total_count_change

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

- DDS_QosPolicyId_t last_policy_id

  Id of the most recent requested incompatible QoS policy.

- DDS_QosPolicyCountSeq policies

  A list of QoS policies and the total number of times each QoS policy was found to be incompatible.

3.29.1 Detailed Description

Status related to the on_offered_incompatible_qos listener methods of the DDS_DataWriter, DDS_Publisher, and DDS_DomainParticipant structures.

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

- dds.h

3.30 DDS_PublicationMatchedStatus Struct Reference

Status related to the on_publication_matched listener methods of the DDS_DataWriter, DDS_Publisher, and DDS_DomainParticipant structures.

Public Attributes

- int total_count

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

- int total_count_change

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

- int current_count

  The current number of DataReaders matched to the DataWriter.

- int current_count_change

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

3.30.1 Detailed Description

Status related to the on_publication_matched listener methods of the DDS_DataWriter, DDS_Publisher, and DDS_DomainParticipant structures.

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

- dds.h

3.31 DDS_Publisher Struct Reference

The DDS_Publisher configures, creates, manages and destroys DDS_DataWriters.

Related Functions

(Note that these are not member functions.)

- DDS_ReturnCode_t DDS_Publisher_enable (DDS_Publisher p)

  Enables the DDS_Publisher.

- DDS_InstanceHandle_t DDS_Publisher_get_instance_handle (DDS_Publisher p)

  This operation returns the InstanceHandle_t that identifies the Publisher.

- DDS_DomainParticipant DDS_Publisher_get_participant (DDS_Publisher p)

  This operation returns the DDS_DomainParticipant this Publisher belongs to.

- DDS_StatusCondition DDS_Publisher_get_statuscondition (DDS_Publisher p)

  This operation allows access to the DDS_StatusCondition associated with the Publisher.

- DDS_StatusMask DDS_Publisher_get_status_changes (DDS_Publisher p)

  *This returns the list of **triggered** communication statuses in the Publisher.*

- DDS_DataWriter DDS_Publisher_create_datawriter (DDS_Publisher p, DDS_Topic a_topic, const DDS_DataWriterQos *qos, DDS_DataWriterListener *a_listener, DDS_StatusMask mask)

  This operation creates a DDS_DataWriter.

- DDS_ReturnCode_t DDS_Publisher_delete_datawriter (DDS_Publisher p, DDS_DataWriter a_datawriter)

  This operation deletes a DataWriter.

- DDS_DataWriter DDS_Publisher_lookup_datawriter (DDS_Publisher p, const char *topic_name)

  *This operation retrieves a previously-created DDS_DataWriter contained in the Publisher, attached to a Topic named **topic_name**.*

- DDS_ReturnCode_t DDS_Publisher_delete_contained_entities (DDS_Publisher p)

  *This operation deletes all the DataWriters created by means of the DDS_Publisher_create_datawriter() operation on the Publisher **p**.*

- DDS_ReturnCode_t DDS_Publisher_set_qos (DDS_Publisher p, const DDS_PublisherQos *qos)

  Sets the DDS_PublisherQos values.

- DDS_ReturnCode_t DDS_Publisher_get_qos (DDS_Publisher p, DDS_PublisherQos *qos)

  *Returns the current DDS_PublisherQos settings held in the Publisher **p**.*

- DDS_ReturnCode_t DDS_Publisher_set_listener (DDS_Publisher p, DDS_PublisherListener *a_listener, DDS_StatusMask mask)

  *Installs a DDS_PublisherListener on Publisher **p**.*

- DDS_ReturnCode_t DDS_Publisher_set_listener_cd (DDS_Publisher p, DDS_PublisherListener_cd *a_listener, DDS_StatusMask mask, void *callback_data)

  *Installs a DDS_PublisherListener_cd on Publisher **p**.*

- DDS_PublisherListener * DDS_Publisher_get_listener (DDS_Publisher p)

  This operation returns the currently installed DDS_PublisherListener.

- DDS_PublisherListener_cd * DDS_Publisher_get_listener_cd (DDS_Publisher p)

  This operation returns the currently installed DDS_PublisherListener_cd.

- DDS_ReturnCode_t DDS_Publisher_suspend_publications (DDS_Publisher p)
- DDS_ReturnCode_t DDS_Publisher_resume_publications (DDS_Publisher p)
- DDS_ReturnCode_t DDS_Publisher_begin_coherent_changes (DDS_Publisher p)
- DDS_ReturnCode_t DDS_Publisher_end_coherent_changes (DDS_Publisher p)
- DDS_Publisher_wait_for_acknowledgements

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

- DDS_ReturnCode_t DDS_Publisher_set_default_datawriter_qos (DDS_Publisher p, const DDS_DataWriterQos *qos)

  Sets the default DDS_DataWriterQos held in the Publisher.

- DDS_ReturnCode_t DDS_Publisher_get_default_datawriter_qos (DDS_Publisher p, struct DDS_DataWriterQos *qos)

  *Provides access to the default DDS_DataWriterQos settings held in the Publisher **p**.*

- DDS_ReturnCode_t DDS_Publisher_copy_from_topic_qos (DDS_Publisher p, struct DDS_DataWriterQos *a_datawriter_qos, const DDS_TopicQos *a_topic_qos)

  *This operation copies the QoS settings in **a_topic_qos** to the corresponding settings in **a_datawriter_qos**.*

3.31.1 Detailed Description

The DDS_Publisher configures, creates, manages and destroys DDS_DataWriters.

3.31.2 Friends And Related Function Documentation

3.31.2.1 DDS_ReturnCode_t DDS_Publisher_begin_coherent_changes (DDS_Publisher *p*) [related]

Not Yet Supported

This operation is not yet implemented.

3.31.2.2 DDS_ReturnCode_t DDS_Publisher_copy_from_topic_qos (DDS_Publisher *p*, struct DDS_DataWriterQos * *a_datawriter_qos*, const DDS_TopicQos * *a_topic_qos*) [related]

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.

3.31.2.3 DDS_DataWriter DDS_Publisher_create_datawriter (DDS_Publisher *p*, DDS_Topic *a_topic*, const DDS_DataWriterQos * *qos*, DDS_DataWriterListener * *a_listener*, DDS_StatusMask *mask*) [related]

This operation creates a DDS_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 DDS_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 DDS_DataWriterListener **a_listener**, is installed at creation time.

The created DataWriter (if not NULL) must be destroyed by a call to DDS_Publisher_delete_datawriter().

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

3.31.2.4 DDS_ReturnCode_t DDS_Publisher_delete_contained_entities (DDS_Publisher *p*) [related]

This operation deletes all the DataWriters created by means of the DDS_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 DDS_DomainParticipant_delete_publisher().

If any of the objects cannot be deleted, this routine will return DDS_RETCODE_PRECONDITION_NOT_MET.

3.31.2.5 DDS_ReturnCode_t DDS_Publisher_delete_datawriter (DDS_Publisher *p*, DDS_DataWriter *a_datawriter*) [related]

This operation deletes a DataWriter.

If the provided DataWriter **a_datawriter** was not created by Publisher **p**, the routine will fail and will return DDS_RETCODE_PRECONDITION_NOT_MET.

3.31.2.6 DDS_ReturnCode_t DDS_Publisher_enable (DDS_Publisher *p*) [related]

Enables the DDS_Publisher.

A Publisher is created either enabled or not based on the DDS_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 DDS_NOT_ENABLED error. DDS_Publisher_enable() may be called on an already enabled Subscriber [it will have no effect].

3.31.2.7 DDS_ReturnCode_t DDS_Publisher_end_coherent_changes (DDS_Publisher *p*) [related]

Not Yet Supported

This operation is not yet implemented.

3.31.2.8 DDS_ReturnCode_t DDS_Publisher_get_default_datawriter_qos (DDS_Publisher *p*, struct DDS_DataWriterQos * *qos*) [related]

Provides access to the default DDS_DataWriterQos settings held in the Publisher **p**.

The provided **qos** argument is populated with the current default qos settings.

3.31.2.9 DDS_PublisherListener * DDS_Publisher_get_listener (DDS_Publisher *p*) [related]

This operation returns the currently installed DDS_PublisherListener.

Note

Because the infrastructure makes a copy of the listener provided in DDS_Publisher_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.

3.31.2.10 DDS_PublisherListener_cd * DDS_Publisher_get_listener_cd (DDS_Publisher *p*) [related]

This operation returns the currently installed DDS_PublisherListener_cd.

Note

Because the infrastructure does not make a copy of the listener provided in DDS_Publisher_set_listener_cd(), the returned structure pointer will match the pointer originally provided. The application should not free the data referenced by the returned pointer.

3.31.2.11 DDS_ReturnCode_t DDS_Publisher_get_qos (DDS_Publisher *p*, DDS_PublisherQos * *qos*) [related]

Returns the current DDS_PublisherQos settings held in the Publisher **p**.

The **qos** parameter is populated with a copy of the current Publisher QoS properties.

Note

The qos structure may contain sequences or strings that are populated with dynamic memory. The caller is responsible for freeing the dynamic memory of these items.
For example, the sequence 'qos->group_data' may have dynamically allocated memory assigned. This can be released by a call to **seq_clear(&qos->group_data)**.

3.31.2.12 DDS_StatusMask DDS_Publisher_get_status_changes (DDS_Publisher *p*) [related]

This returns the list of **triggered** communication statuses in the Publisher.

If the Publisher is not enabled, all statuses will be **untriggered**. The list returned by **get_status_changes** may be empty. The list of statuses returned by **get_status_changes** operation contains statuses that are triggered on the Publisher itself and does not include statuses from contained **DataWriter** objects.

3.31.2.13 DDS_StatusCondition DDS_Publisher_get_statuscondition (DDS_Publisher *p*) [related]

This operation allows access to the DDS_StatusCondition associated with the Publisher.

The returned condition can be added to a DDS_WaitSet.

3.31.2.14 DDS_DataWriter DDS_Publisher_lookup_datawriter (DDS_Publisher *p,* const char * *topic_name*) [related]

This operation retrieves a previously-created DDS_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**.

3.31.2.15 DDS_ReturnCode_t DDS_Publisher_resume_publications (DDS_Publisher *p*) [related]

Not Yet Supported

This operation is not yet implemented.

3.31.2.16 DDS_ReturnCode_t DDS_Publisher_set_default_datawriter_qos (DDS_Publisher *p,* const DDS_DataWriterQos * *qos*) [related]

Sets the default DDS_DataWriterQos held in the Publisher.

This default qos will be used during subsequent calls to DDS_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, DDS_-INCONSISTENT_POLICY will be returned, and no changes will be made to the Publisher.

3.31.2.17 DDS_ReturnCode_t DDS_Publisher_set_listener (DDS_Publisher *p,* DDS_PublisherListener * *a_listener,* DDS_StatusMask *mask*) [related]

Installs a DDS_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.

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

3.31.2.18 DDS_ReturnCode_t DDS_Publisher_set_listener_cd (DDS_Publisher *p*, DDS_PublisherListener_cd * *a_listener*, DDS_StatusMask *mask*, void * *callback_data*) [related]

Installs a DDS_PublisherListener_cd on Publisher **p**.

Only one listener may be attached to a Publisher at a time. A call to **set_listener_cd()** will replace any current listener with **a_listener**.

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

The infrastructure will not make an internal copy of the listener structure which means that it must be persisted by the application.

3.31.2.19 DDS_ReturnCode_t DDS_Publisher_set_qos (DDS_Publisher *p*, const DDS_PublisherQos * *qos*) [related]

Sets the DDS_PublisherQos values.

These QoS values affect the behavior of the DDS_Publisher.

3.31.2.20 DDS_ReturnCode_t DDS_Publisher_suspend_publications (DDS_Publisher *p*) [related]

Not Yet Supported

This operation is not yet implemented.

3.31.2.21 DDS_Publisher_wait_for_acknowledgements [related]

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 struct was generated from the following file:

- dds.h

3.32 DDS_PublisherListener Struct Reference

The DDS_PublisherListener provides asynchronous notification of DDS_Publisher events.

Public Attributes

- void(∗ on_offered_deadline_missed)(DDS_DataWriter writer, DDS_OfferedDeadlineMissedStatus status)
- void(∗ on_offered_incompatible_qos)(DDS_DataWriter writer, DDS_OfferedIncompatibleQosStatus status)
- void(∗ on_liveliness_lost)(DDS_DataWriter writer, DDS_LivelinessLostStatus status)
- void(∗ on_publication_matched)(DDS_DataWriter writer, DDS_PublicationMatchedStatus status)

3.32.1 Detailed Description

The DDS_PublisherListener provides asynchronous notification of DDS_Publisher events. This listener can be installed during Publisher creation DDS_DomainParticipant_create_publisher(), as well as by calling DDS_Publisher_set_listener().

Note

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

3.32.2 Member Data Documentation

3.32.2.1 void(∗ DDS_PublisherListener::on_liveliness_lost)(DDS_DataWriter writer, DDS_LivelinessLostStatus status)

on_liveliness_lost() is 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 DDS_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.

3.32.2.2 void(∗ DDS_PublisherListener::on_offered_deadline_missed)(DDS_DataWriter writer, DDS_OfferedDeadlineMissedStatus status)

on_offered_deadline_missed() is 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 DDS_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.

3.32.2.3 void(* DDS_PublisherListener::on_offered_incompatible_qos)(DDS_DataWriter writer, DDS_OfferedIncompatibleQosStatus status)

on_offered_incompatible_qos() is 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 DDS_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.

3.32.2.4 void(* DDS_PublisherListener::on_publication_matched)(DDS_DataWriter writer, DDS_PublicationMatchedStatus status)

on_publication_matched() is 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 DDS_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 struct was generated from the following file:

- dds.h

3.33 DDS_PublisherListener_cd Struct Reference

The DDS_PublisherListener_cd provides asynchronous notification of DDS_Publisher events with addition-all callback data.

Public Attributes

- void(∗ on_offered_deadline_missed)(struct DDS_PublisherListener_cd ∗self, DDS_DataWriter writer, DDS_OfferedDeadlineMissedStatus status, void ∗callback_data)
- void(∗ on_offered_incompatible_qos)(struct DDS_PublisherListener_cd ∗self, DDS_DataWriter writer, DDS_OfferedIncompatibleQosStatus status, void ∗callback_data)
- void(∗ on_liveliness_lost)(struct DDS_PublisherListener_cd ∗self, DDS_DataWriter writer, DDS_-LivelinessLostStatus status, void ∗callback_data)
- void(∗ on_publication_matched)(struct DDS_PublisherListener_cd ∗self, DDS_DataWriter writer, DDS_PublicationMatchedStatus status, void ∗callback_data)

3.33.1 Detailed Description

The DDS_PublisherListener_cd provides asynchronous notification of DDS_Publisher events with addition-all callback data. This listener can be installed by calling DDS_Publisher_set_listener_cd().

Note

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

3.33.2 Member Data Documentation

3.33.2.1 void(∗ DDS_PublisherListener_cd::on_liveliness_lost)(struct DDS_PublisherListener_cd ∗self, DDS_DataWriter writer, DDS_LivelinessLostStatus status, void ∗callback_data)

on_liveliness_lost() is 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 DDS_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.

3.33.2.2 void(∗ DDS_PublisherListener_cd::on_offered_deadline_missed)(struct DDS_-PublisherListener_cd ∗self, DDS_DataWriter writer, DDS_OfferedDeadlineMissedStatus status, void ∗callback_data)

on_offered_deadline_missed() is 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 DDS_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.

3.33.2.3 void(∗ DDS_PublisherListener_cd::on_offered_incompatible_qos)(struct DDS_PublisherListener_cd ∗self, DDS_DataWriter writer, DDS_OfferedIncompatibleQosStatus status, void ∗callback_data)

on_offered_incompatible_qos() is 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 DDS_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.

3.33.2.4 void(∗ DDS_PublisherListener_cd::on_publication_matched)(struct DDS_PublisherListener_cd ∗self, DDS_DataWriter writer, DDS_PublicationMatchedStatus status, void ∗callback_data)

on_publication_matched() is 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 DDS_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 struct was generated from the following file:

- dds.h

3.34 DDS_PublisherQos Struct Reference

Structure that holds DDS_Publisher Quality of Service policies.

Public Attributes

- DDS_PresentationQosPolicy presentation

  Controls the presentation of groups of changes.

- DDS_PartitionQosPolicy partition

  Establishes a logical data partition.

- DDS_GroupDataQosPolicy group_data

  A sequence of octets associated with the Publisher.

- DDS_EntityFactoryQosPolicy entity_factory

  Controls the behavior of the Publisher_create_datawriter() operation.

3.34.1 Detailed Description

Structure that holds DDS_Publisher Quality of Service policies.

See also

- DDS_Publisher_set_qos()
- DDS_Publisher_get_qos()
- DDS_DomainParticipant_create_publisher()
- DDS_DomainParticipant_set_default_publisher_qos()
- DDS_DomainParticipant_get_default_publisher_qos()

3.34.2 Member Data Documentation

3.34.2.1 DDS_PartitionQosPolicy DDS_PublisherQos::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.

3.34.2.2 DDS_PresentationQosPolicy DDS_PublisherQos::presentation

Controls the presentation of groups of changes.

See also

- DDS_Publisher_begin_coherent_changes()
- DDS_Publisher_end_coherent_changes()
- DDS_Subscriber_begin_access()
- DDS_Subscriber_end_access()

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

- dds.h

3.35 DDS_QueryCondition Struct Reference

A DDS_QueryCondition is a specialized DDS_ReadCondition which includes a filter.

Related Functions

(Note that these are not member functions.)

- unsigned char DDS_QueryCondition_get_trigger_value (DDS_QueryCondition qc)
  *This routine returns the current value of the **trigger_value** in **qc**.*
- DDS_DataReader DDS_QueryCondition_get_datareader (DDS_QueryCondition qc)
  This routine returns the single DDS_DataReader associated with this QueryCondition.
- DDS_SampleStateKind DDS_QueryCondition_get_sample_state_mask (DDS_QueryCondition qc)
  This routine returns the current value of the sample_state in this QueryCondition.
- DDS_ViewStateKind DDS_QueryCondition_get_view_state_mask (DDS_QueryCondition qc)
  This routine returns the current value of the view_state in this QueryCondition.
- DDS_InstanceStateKind DDS_QueryCondition_get_instance_state_mask (DDS_QueryCondition qc)
  This routine returns the current value of the instance_state in this QueryCondition.
- const char * DDS_QueryCondition_get_query_expression (DDS_QueryCondition qc)
  This routine returns the SQL query expression of this QueryCondition.
- DDS_ReturnCode_t DDS_QueryCondition_get_query_parameters (DDS_QueryCondition qc, DDS_StringSeq *seq)
  This routine returns the parameters to the SQL query expression of this QueryCondition.
- DDS_ReturnCode_t DDS_QueryCondition_set_query_parameters (DDS_QueryCondition qc, DDS_StringSeq *parameters)
  This routine sets the parameters to the SQL query expression of this QueryCondition.

3.35.1 Detailed Description

A DDS_QueryCondition is a specialized DDS_ReadCondition which includes a filter. The **trigger_value** is driven by the data available, after applying the filter, in the associated DataReader.

Not Yet Supported

CoreDX DS does not yet implement QueryConditions as a trigger for a WaitSet.

3.35.2 Friends And Related Function Documentation

3.35.2.1 const char * DDS_QueryCondition_get_query_expression (DDS_QueryCondition *qc*) [related]

This routine returns the SQL query expression of this QueryCondition.

The query expression is an SQL syntax conditional expression that is provided when the QueryCondition is created.

3.35.2.2 DDS_ReturnCode_t DDS_QueryCondition_get_query_parameters (DDS_QueryCondition *qc,* DDS_StringSeq * *seq*) [related]

This routine returns the parameters to the SQL query expression of this QueryCondition.

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

DDS_DataReader_create_querycondition()
DDS_QueryCondition_set_query_parameters()

3.35.2.3 unsigned char DDS_QueryCondition_get_trigger_value (DDS_QueryCondition *qc*) [related]

This routine returns the current value of the **trigger_value** in **qc**.

A non-zero return value indicates that the **trigger_value** is TRUE.

A zero return value indicates that the **trigger_value** is FALSE.

3.35.2.4 DDS_ReturnCode_t DDS_QueryCondition_set_query_parameters (DDS_QueryCondition *qc,* DDS_StringSeq * *parameters*) [related]

This routine sets the parameters to the SQL query expression of this QueryCondition.

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

DDS_DataReader_create_querycondition()

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

- dds.h

3.36 DDS_ReadCondition Struct Reference

A DDS_ReadCondition is a specialized DDS_Condition associated with a DDS_DataReader.

Related Functions

(Note that these are not member functions.)

- unsigned char DDS_ReadCondition_get_trigger_value (DDS_ReadCondition rc)
  *This routine returns the current value of the **trigger_value** in **rc**.*
- DDS_DataReader DDS_ReadCondition_get_datareader (DDS_ReadCondition rc)
  This routine returns the single DDS_DataReader associated with this ReadCondition.
- DDS_SampleStateKind DDS_ReadCondition_get_sample_state_mask (DDS_ReadCondition rc)
  This routine returns the current value of the sample_state in this ReadCondition.
- DDS_ViewStateKind DDS_ReadCondition_get_view_state_mask (DDS_ReadCondition rc)
  This routine returns the current value of the view_state in this ReadCondition.
- DDS_InstanceStateKind DDS_ReadCondition_get_instance_state_mask (DDS_ReadCondition rc)
  This routine returns the current value of the instance_state in this ReadCondition.

3.36.1 Detailed Description

A DDS_ReadCondition is a specialized DDS_Condition associated with a DDS_DataReader. The **trigger_value** is driven by the data available in the associated DataReader. A ReadCondition is obtained by calling the DDS_DataReader_create_readcondition() function. When the ReadCondition is no longer needed, it should be destroyed by a call to DDS_DataReader_delete_readcondition().

3.36.2 Friends And Related Function Documentation

3.36.2.1 unsigned char DDS_ReadCondition_get_trigger_value (DDS_ReadCondition *rc*) [related]

This routine returns the current value of the **trigger_value** in **rc**.

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 struct was generated from the following file:

- dds.h

3.37 DDS_RequestedDeadlineMissedStatus Struct Reference

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

Public Attributes

- int total_count

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

- int total_count_change

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

- DDS_InstanceHandle_t last_instance_handle

  Handle identifying the most recent instance whose deadline was missed.

3.37.1 Detailed Description

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

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

- dds.h

3.38 DDS_RequestedIncompatibleQosStatus Struct Reference

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

Public Attributes

- int total_count

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

- int total_count_change

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

- DDS_QosPolicyId_t last_policy_id

  Handle identifying the most recent QoS policy detected to be incompatible.

- DDS_QosPolicyCountSeq policies

  A list of QoS policies and the total number of times each QoS policy was found to be incompatible.

3.38.1 Detailed Description

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

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

- dds.h

3.39 DDS_SampleInfo Struct Reference

The SampleInfo structure contains information associated with each sample.

Public Attributes

- DDS_SampleStateKind sample_state

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

- DDS_ViewStateKind view_state

  Associated instance has/has not been seen before.

- DDS_InstanceStateKind 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.

- DDS_Time_t source_timestamp

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

- DDS_Time_t reception_timestamp

  The time the sample was received by the DataReader.

- DDS_InstanceHandle_t instance_handle

  The handle that locally identifies the associated instance.

- DDS_InstanceHandle_t publication_handle

  The local handle of the source DataWriter.

- 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.).

- 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.).

- int sample_rank

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

- int generation_rank

  The generation difference of this sample and the most recent sample in the collection.

- int absolute_generation_rank

  The generation difference between this sample and the most recent sample.

- unsigned char valid_data

  Set to true (non-zero) if the associated DataSample contains data.

3.39.1 Detailed Description

The SampleInfo structure contains information associated with each sample.

3.39.2 Member Data Documentation

3.39.2.1 int DDS_SampleInfo::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.

3.39.2.2 int DDS_SampleInfo::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.

3.39.2.3 DDS_InstanceStateKind DDS_SampleInfo::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) ...
```

3.39.2.4 DDS_SampleStateKind DDS_SampleInfo::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.

3.39.2.5 unsigned char DDS_SampleInfo::valid_data

Set to true (non-zero) 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).

3.39.2.6 DDS_ViewStateKind DDS_SampleInfo::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 struct was generated from the following file:

- dds.h

3.40 DDS_SampleLostStatus Struct Reference

Status related to the on_sample_lost listener methods of the DDS_DataReader, DDS_Subscriber, and DDS_-DomainParticipant structures.

Public Attributes

- int total_count

  Cummulative count of all samples lost under the Topic.

- int total_count_change

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

3.40.1 Detailed Description

Status related to the on_sample_lost listener methods of the DDS_DataReader, DDS_Subscriber, and DDS_-DomainParticipant structures.

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

- dds.h

3.41 DDS_SampleRejectedStatus Struct Reference

Status related to the on_sample_rejected listener methods of the DDS_DataReader, DDS_Subscriber, and DDS_DomainParticipant structures.

Public Attributes

- int total_count

  Cummulative count of samples rejected by the DataReader.

- int total_count_change

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

- DDS_SampleRejectedStatusKind last_reason

  The reason for rejecting the most recently rejected sample.

- DDS_InstanceHandle_t last_instance_handle

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

3.41.1 Detailed Description

Status related to the on_sample_rejected listener methods of the DDS_DataReader, DDS_Subscriber, and DDS_DomainParticipant structures.

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

- dds.h

3.42 DDS_StatusCondition Struct Reference

A DDS_StatusCondition is a condition associated with an Entity.

Related Functions

(Note that these are not member functions.)

- unsigned char DDS_StatusCondition_get_trigger_value (DDS_StatusCondition sc)

  *This routine returns the current value of the **trigger_value** in **gc**.*

- DDS_StatusMask DDS_StatusCondition_get_enabled_statuses (DDS_StatusCondition sc)

  This routine returns the statuses which are enabled in this StatusCondition.

- DDS_ReturnCode_t DDS_StatusCondition_set_enabled_statuses (DDS_StatusCondition sc, DDS_StatusMask mask)

  This routine sets the statuses which are enabled in this StatusCondition.

- DDS_Entity DDS_StatusCondition_get_entity (DDS_StatusCondition sc)

  This routine returns the single entity associated with this StatusCondition.

3.42.1 Detailed Description

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

3.42.2 Friends And Related Function Documentation

3.42.2.1 DDS_StatusMask DDS_StatusCondition_get_enabled_statuses (DDS_StatusCondition *sc*) [related]

This routine returns the statuses which are enabled in this StatusCondition.

The statuses are returned as a bitmask.

3.42.2.2 unsigned char DDS_StatusCondition_get_trigger_value (DDS_StatusCondition *sc*) [related]

This routine returns the current value of the **trigger_value** in **gc**.

A non-zero return value indicates that the **trigger_value** is TRUE.

A zero return value indicates that the **trigger_value** is FALSE.

3.42.2.3 DDS_ReturnCode_t DDS_StatusCondition_set_enabled_statuses (DDS_StatusCondition *sc*, DDS_StatusMask *mask*) [related]

This routine sets the statuses which are enabled in this StatusCondition.

The statuses are provided as a bitmask.

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

- dds.h

3.43 DDS_Subscriber Struct Reference

The DDS_Subscriber configures, creates, manages and destroys DDS_DataReaders.

Related Functions

(Note that these are not member functions.)

- DDS_ReturnCode_t DDS_Subscriber_enable (DDS_Subscriber s)

  Enables the DDS_Subscriber.

- DDS_InstanceHandle_t DDS_Subscriber_get_instance_handle (DDS_Subscriber s)

  This operation returns the InstanceHandle_t that identifies the Subscriber.

- DDS_DomainParticipant DDS_Subscriber_get_participant (DDS_Subscriber s)

  This operation returns the DDS_DomainParticipant this Subscriber belongs to.

- DDS_StatusCondition DDS_Subscriber_get_statuscondition (DDS_Subscriber s)

  This operation allows access to the DDS_StatusCondition associated with the Subscriber.

- DDS_StatusMask DDS_Subscriber_get_status_changes (DDS_Subscriber s)

  *This returns the list of **triggered** communication statuses in the Subscriber.*

- DDS_DataReader DDS_Subscriber_create_datareader (DDS_Subscriber s, DDS_TopicDescription a_topic, DDS_DataReaderQos *qos, DDS_DataReaderListener *a_listener, DDS_StatusMask mask)

  This operation creates a DDS_DataReader.

- DDS_ReturnCode_t DDS_Subscriber_delete_datareader (DDS_Subscriber s, DDS_DataReader a_datareader)

  This operation deletes a DataReader.

- DDS_ReturnCode_t DDS_Subscriber_delete_contained_entities (DDS_Subscriber s)

  *This operation deletes all the DataReaders created by means of the DDS_Subscriber_create_datareader() operation on the Subscriber **s**.*

- DDS_DataReader DDS_Subscriber_lookup_datareader (DDS_Subscriber s, const char *topic_name)

  *This operation retrieves a previously-created DDS_DataReader contained in the Subscriber, attached to a Topic named **topic_name**.*

- DDS_ReturnCode_t DDS_Subscriber_get_datareaders (DDS_Subscriber s, DDS_DataReaderSeq *readers, DDS_SampleStateMask sample_states, DDS_ViewStateMask view_states, DDS_InstanceStateMask instance_states)

*This operation allows the application to access DataReader objects that contain samples with the specified **sample_states**, **view_states**, and **instance_states**.*

- DDS_ReturnCode_t DDS_Subscriber_notify_datareaders (DDS_Subscriber s)

  *This operation invokes the DDS_DataReaderListener **on_data_available** operation on any contained DataReader entities with a DATA_AVAILABLE status that is considered changed.*

- DDS_ReturnCode_t DDS_Subscriber_set_qos (DDS_Subscriber s, DDS_SubscriberQos *qos)

  Sets the DDS_SubscriberQos values.

- DDS_ReturnCode_t DDS_Subscriber_get_qos (DDS_Subscriber s, DDS_SubscriberQos *qos)

  *Returns the current DDS_SubscriberQos settings held in the Subscriber **s**.*

- DDS_ReturnCode_t DDS_Subscriber_set_listener (DDS_Subscriber s, DDS_SubscriberListener *a_listener, DDS_StatusMask mask)

  *Installs a DDS_SubscriberListener on Subscriber **s**.*

- DDS_ReturnCode_t DDS_Subscriber_set_listener_cd (DDS_Subscriber s, DDS_SubscriberListener_cd *listener_cd, DDS_StatusMask mask, void *callback_data)

  *Installs a DDS_SubscriberListener_cd on Subscriber **s**.*

- DDS_SubscriberListener * DDS_Subscriber_get_listener (DDS_Subscriber s)

  This operation returns the currently installed DDS_SubscriberListener.

- DDS_SubscriberListener_cd * DDS_Subscriber_get_listener_cd (DDS_Subscriber s)

  This operation returns the currently installed DDS_SubscriberListener_cd.

- DDS_ReturnCode_t DDS_Subscriber_begin_access (DDS_Subscriber s)

  *This operation indicates that the application is about to access the data samples in any of the DataReader objects contained in the Subscriber **s**.*

- DDS_ReturnCode_t DDS_Subscriber_end_access (DDS_Subscriber s)

  This operation closes a corresponding DDS_Subscriber_begin_access().

- DDS_ReturnCode_t DDS_Subscriber_set_default_datareader_qos (DDS_Subscriber s, const DDS_DataReaderQos *qos)

  Sets the default DDS_DataReaderQos held in the Subscriber.

- DDS_ReturnCode_t DDS_Subscriber_get_default_datareader_qos (DDS_Subscriber s, DDS_DataReaderQos *qos)

  *Provides access to the default DDS_DataReaderQos settings held in the Subscriber **s**.*

- DDS_ReturnCode_t DDS_Subscriber_copy_from_topic_qos (DDS_Subscriber s, DDS_DataReaderQos *a_datareader_qos, DDS_TopicQos *a_topic_qos)

*This operation copies the QoS settings in **a_topic_qos** to the corresponding settings in **a_datareader_qos**.*

3.43.1 Detailed Description

The DDS_Subscriber configures, creates, manages and destroys DDS_DataReaders.

3.43.2 Friends And Related Function Documentation

3.43.2.1 DDS_ReturnCode_t DDS_Subscriber_begin_access (DDS_Subscriber *s*) [related]

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.]

3.43.2.2 DDS_ReturnCode_t DDS_Subscriber_copy_from_topic_qos (DDS_Subscriber *s*, DDS_DataReaderQos * *a_datareader_qos*, DDS_TopicQos * *a_topic_qos*) [related]

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.

3.43.2.3 DDS_DataReader DDS_Subscriber_create_datareader (DDS_Subscriber *s*, DDS_TopicDescription *a_topic*, DDS_DataReaderQos * *qos*, DDS_DataReaderListener * *a_listener*, DDS_StatusMask *mask*) [related]

This operation creates a DDS_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 DDS_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 DDS_DataReaderListener **a_listener**, is installed at creation time.

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

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

3.43.2.4 DDS_ReturnCode_t DDS_Subscriber_delete_contained_entities (DDS_Subscriber *s*) [related]

This operation deletes all the DataReaders created by means of the DDS_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 DDS_DomainParticipant_delete_subscriber().

If any of the objects cannot be deleted, this routine will return DDS_RETCODE_PRECONDITION_NOT_MET.

3.43.2.5 DDS_ReturnCode_t DDS_Subscriber_delete_datareader (DDS_Subscriber *s*, DDS_DataReader *a_datareader*) [related]

This operation deletes a DataReader.

If the provided DataReader **a_datareader** was not created by Subscriber **s**, the routine will fail and will return DDS_RETCODE_PRECONDITION_NOT_MET. If the indicated DataReader has any outstanding DDS_ReadCondition or DDS_QueryCondition objects the routine will fail and will return DDS_RETCODE_PRECONDITION_NOT_MET. If the indicated DataReader has any outstanding 'loans' (from read() or take() operations), the routine will fail and will return DDS_RETCODE_PRECONDITION_NOT_MET.

3.43.2.6 DDS_ReturnCode_t DDS_Subscriber_enable (DDS_Subscriber *s*) [related]

Enables the DDS_Subscriber.

A Subscriber is created either enabled or not based on the DDS_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 DDS_NOT_ENABLED error. DDS_Subscriber_enable() may be called on an already enabled Subscriber [it will have no effect].

3.43.2.7 DDS_ReturnCode_t DDS_Subscriber_get_datareaders (DDS_Subscriber *s*, DDS_DataReaderSeq * *readers*, DDS_SampleStateMask *sample_states*, DDS_ViewStateMask *view_states*, DDS_InstanceStateMask *instance_states*) [related]

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.

3.43.2.8 DDS_ReturnCode_t DDS_Subscriber_get_default_datareader_qos (DDS_Subscriber *s*, DDS_DataReaderQos * *qos*) [related]

Provides access to the default DDS_DataReaderQos settings held in the Subscriber **s**.

The provided **qos** argument is populated with the current default qos settings.

3.43.2.9 DDS_SubscriberListener * DDS_Subscriber_get_listener (DDS_Subscriber *s*) [related]

This operation returns the currently installed DDS_SubscriberListener.

Note

Because the infrastructure makes a copy of the listener provided in DDS_Subscriber_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.

3.43.2.10 DDS_SubscriberListener_cd * DDS_Subscriber_get_listener_cd (DDS_Subscriber *s*) [related]

This operation returns the currently installed DDS_SubscriberListener_cd.

Note

Because the infrastructure does not make a copy of the listener provided in DDS_Subscriber_set_listener_cd(), the returned structure pointer will match the pointer originally provided. The application should not free the data referenced by the returned pointer.

3.43.2.11 DDS_ReturnCode_t DDS_Subscriber_get_qos (DDS_Subscriber *s*, DDS_SubscriberQos * *qos*) [related]

Returns the current DDS_SubscriberQos settings held in the Subscriber **s**.

The **qos** parameter is populated with a copy of the current Subscriber QoS properties.

Note

The qos structure may contain sequences or strings that are populated with dynamic memory. The caller is responsible for freeing the dynamic memory of these items.
For example, the sequence 'qos->group_data' may have dynamically allocated memory assigned. This can be released by a call to **seq_clear(&qos->group_data)**.

3.43.2.12 DDS_StatusMask DDS_Subscriber_get_status_changes (DDS_Subscriber *s*) [related]

This returns the list of **triggered** communication statuses in the Subscriber.

If the Subscriber is not enabled, all statuses will be **untriggered**. The list returned by **get_status_changes** may be empty. The list of statuses returned by **get_status_changes** operation contains statuses that are triggered on the Subscriber itself and does not include statuses from contained **DataReader** objects.

3.43.2.13 DDS_StatusCondition DDS_Subscriber_get_statuscondition (DDS_Subscriber *s*) [related]

This operation allows access to the DDS_StatusCondition associated with the Subscriber.

The returned condition can be added to a DDS_WaitSet.

3.43.2.14 DDS_DataReader DDS_Subscriber_lookup_datareader (DDS_Subscriber *s*, const char * *topic_name*) [related]

This operation retrieves a previously-created DDS_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.

3.43.2.15 DDS_ReturnCode_t DDS_Subscriber_set_default_datareader_qos (DDS_Subscriber *s*, const DDS_DataReaderQos * *qos*) [related]

Sets the default DDS_DataReaderQos held in the Subscriber.

This default qos will be used during subsequent calls to DDS_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, DDS_-INCONSISTENT_POLICY will be returned, and no changes will be made to the Subscriber.

3.43.2.16 DDS_ReturnCode_t DDS_Subscriber_set_listener (DDS_Subscriber *s*, DDS_SubscriberListener * *a_listener*, DDS_StatusMask *mask*) [related]

Installs a DDS_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.

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

3.43.2.17 DDS_ReturnCode_t DDS_Subscriber_set_listener_cd (DDS_Subscriber *s*, DDS_SubscriberListener_cd * *listener_cd*, DDS_StatusMask *mask*, void * *callback_data*) [related]

Installs a DDS_SubscriberListener_cd on Subscriber **s**.

Only one listener may be attached to a Subscriber at a time. A call to **set_listener_cd()** will replace any current listener with **a_listener**.

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

The infrastructure will not make an internal copy of the listener_cd structure which means that it must be persisted by the application.

3.43.2.18 DDS_ReturnCode_t DDS_Subscriber_set_qos (DDS_Subscriber *s*, DDS_SubscriberQos * *qos*) [related]

Sets the DDS_SubscriberQos values.

These QoS values affect the behavior of the DDS_Subscriber.

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

- dds.h

3.44 DDS_SubscriberListener Struct Reference

The DDS_SubscriberListener provides asynchronous notification of DDS_Subscriber events.

Public Attributes

- void(* on_requested_deadline_missed)(DDS_DataReader the_reader, DDS_RequestedDeadlineMissedStatus status)
- void(* on_requested_incompatible_qos)(DDS_DataReader the_reader, DDS_RequestedIncompatibleQosStatus status)
- void(* on_sample_rejected)(DDS_DataReader the_reader, DDS_SampleRejectedStatus status)
- void(* on_liveliness_changed)(DDS_DataReader the_reader, DDS_LivelinessChangedStatus status)
- void(* on_data_available)(DDS_DataReader the_reader)
- void(* on_subscription_matched)(DDS_DataReader the_reader, DDS_SubscriptionMatchedStatus status)
- void(* on_sample_lost)(DDS_DataReader the_reader, DDS_SampleLostStatus status)
- void(* on_data_on_readers)(DDS_Subscriber the_subscriber)

3.44.1 Detailed Description

The DDS_SubscriberListener provides asynchronous notification of DDS_Subscriber events. This listener can be installed during Subscriber creation, DDS_DomainParticipant_create_subscriber() as well as by calling DDS_Subscriber_set_listener().

Note

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

3.44.2 Member Data Documentation

3.44.2.1 void(* DDS_SubscriberListener::on_data_available)(DDS_DataReader the_reader)

on_data_available() is 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 DDS_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.

3.44.2.2 void(* DDS_SubscriberListener::on_data_on_readers)(DDS_Subscriber the_subscriber)

on_data_on_readers() is 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.

3.44.2.3 void(* DDS_SubscriberListener::on_liveliness_changed)(DDS_DataReader the_reader, DDS_LivelinessChangedStatus status)

on_liveliness_changed() is 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 DDS_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.

3.44.2.4 void(* DDS_SubscriberListener::on_requested_deadline_missed)(DDS_DataReader the_reader, DDS_RequestedDeadlineMissedStatus status)

on_requested_deadline_missed() is 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 DDS_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.

3.44.2.5 void(* DDS_SubscriberListener::on_requested_incompatible_qos)(DDS_DataReader the_reader, DDS_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 DDS_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.

3.44.2.6 void(* DDS_SubscriberListener::on_sample_lost)(DDS_DataReader the_reader, DDS_SampleLostStatus status)

on_sample_lost() is 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 DDS_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.

3.44.2.7 void(* DDS_SubscriberListener::on_sample_rejected)(DDS_DataReader the_reader, DDS_SampleRejectedStatus status)

on_sample_rejected() is 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 DDS_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.

3.44.2.8 void(* DDS_SubscriberListener::on_subscription_matched)(DDS_DataReader the_reader, DDS_SubscriptionMatchedStatus status)

on_subscription_matched() is 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 DDS_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 struct was generated from the following file:

- dds.h

3.45 DDS_SubscriberListener_cd Struct Reference

The DDS_SubscriberListener_cd provides asynchronous notification of DDS_Subscriber events with additional callback data.

Public Attributes

- void(∗ on_requested_deadline_missed)(struct DDS_SubscriberListener_cd ∗self, DDS_DataReader the_reader, DDS_RequestedDeadlineMissedStatus status, void ∗callback_data)
- void(∗ on_requested_incompatible_qos)(struct DDS_SubscriberListener_cd ∗self, DDS_DataReader the_reader, DDS_RequestedIncompatibleQosStatus status, void ∗callback_data)
- void(∗ on_sample_rejected)(struct DDS_SubscriberListener_cd ∗self, DDS_DataReader the_reader, DDS_SampleRejectedStatus status, void ∗callback_data)
- void(∗ on_liveliness_changed)(struct DDS_SubscriberListener_cd ∗self, DDS_DataReader the_reader, DDS_LivelinessChangedStatus status, void ∗callback_data)
- void(∗ on_data_available)(struct DDS_SubscriberListener_cd ∗self, DDS_DataReader the_reader, void ∗callback_data)
- void(∗ on_subscription_matched)(struct DDS_SubscriberListener_cd ∗self, DDS_DataReader the_reader, DDS_SubscriptionMatchedStatus status, void ∗callback_data)
- void(∗ on_sample_lost)(struct DDS_SubscriberListener_cd ∗self, DDS_DataReader the_reader, DDS_SampleLostStatus status, void ∗callback_data)
- void(∗ on_data_on_readers)(struct DDS_SubscriberListener_cd ∗self, DDS_Subscriber the_subscriber, void ∗callback_data)

3.45.1 Detailed Description

The DDS_SubscriberListener_cd provides asynchronous notification of DDS_Subscriber events with additional callback data. This listener can be installed by calling DDS_Subscriber_set_listener_cd().

Note

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

3.45.2 Member Data Documentation

3.45.2.1 void(∗ DDS_SubscriberListener_cd::on_data_available)(struct DDS_SubscriberListener_cd ∗self, DDS_DataReader the_reader, void ∗callback_data)

on_data_available() is 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 DDS_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.

3.45.2.2 void(* DDS_SubscriberListener_cd::on_data_on_readers)(struct DDS_SubscriberListener_cd *self, DDS_Subscriber the_subscriber, void *callback_data)

on_data_on_readers() is 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.

3.45.2.3 void(* DDS_SubscriberListener_cd::on_liveliness_changed)(struct DDS_SubscriberListener_cd *self, DDS_DataReader the_reader, DDS_LivelinessChangedStatus status, void *callback_data)

on_liveliness_changed() is 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 DDS_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.

3.45.2.4 void(* DDS_SubscriberListener_cd::on_requested_deadline_missed)(struct DDS_SubscriberListener_cd *self, DDS_DataReader the_reader, DDS_RequestedDeadlineMissedStatus status, void *callback_data)

on_requested_deadline_missed() is 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 DDS_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.

3.45.2.5 void(* DDS_SubscriberListener_cd::on_requested_incompatible_qos)(struct DDS_SubscriberListener_cd *self, DDS_DataReader the_reader, DDS_RequestedIncompatibleQosStatus status, void *callback_data)

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 DDS_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.

3.45.2.6 void(* DDS_SubscriberListener_cd::on_sample_lost)(struct DDS_SubscriberListener_cd *self, DDS_DataReader the_reader, DDS_SampleLostStatus status, void *callback_data)

on_sample_lost() is 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 DDS_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.

3.45.2.7 void(* DDS_SubscriberListener_cd::on_sample_rejected)(struct DDS_SubscriberListener_cd *self, DDS_DataReader the_reader, DDS_SampleRejectedStatus status, void *callback_data)

on_sample_rejected() is 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 DDS_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.

3.45.2.8 void(* DDS_SubscriberListener_cd::on_subscription_matched)(struct DDS_SubscriberListener_cd *self, DDS_DataReader the_reader, DDS_SubscriptionMatchedStatus status, void *callback_data)

on_subscription_matched() is 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 DDS_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 struct was generated from the following file:

- dds.h

3.46 DDS_SubscriberQos Struct Reference

Structure that holds DDS_Subscriber Quality of Service policies.

Public Attributes

- DDS_PresentationQosPolicy presentation

  Controls the presentation of groups of changes.

- DDS_PartitionQosPolicy partition

  Establishes a logical data partition.

- DDS_GroupDataQosPolicy group_data

  A sequence of octets associated with the Publisher.

- DDS_EntityFactoryQosPolicy entity_factory

  Controls the behavior of the Subsriber_create_datareader() operation.

3.46.1 Detailed Description

Structure that holds DDS_Subscriber Quality of Service policies.

See also

- DDS_Subscriber_set_qos()
- DDS_Subscriber_get_qos()
- DDS_DomainParticipant_create_subscriber()
- DDS_DomainParticipant_set_default_subscriber_qos()
- DDS_DomainParticipant_get_default_subscriber_qos()

3.46.2 Member Data Documentation

3.46.2.1 DDS_PartitionQosPolicy DDS_SubscriberQos::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.

3.46.2.2 DDS_PresentationQosPolicy DDS_SubscriberQos::presentation

Controls the presentation of groups of changes.

See also

- DDS_Publisher_begin_coherent_changes()
- DDS_Publisher_end_coherent_changes()
- DDS_Subscriber_begin_access()
- DDS_Subscriber_end_access()

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

- dds.h

3.47 DDS_SubscriptionMatchedStatus Struct Reference

Status related to the on_subscription_matched listener methods of the DDS_DataReader, DDS_Subscriber, and DDS_DomainParticipant structures.

Public Attributes

- int total_count

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

- int total_count_change

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

- int current_count

  The current number of DataWriters matched to the DataReader.

- int current_count_change

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

3.47.1 Detailed Description

Status related to the on_subscription_matched listener methods of the DDS_DataReader, DDS_Subscriber, and DDS_DomainParticipant structures.

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

- dds.h

3.48 DDS_Topic Struct Reference

DDS_Topic is the basic description of data to be published or subscribed.

Related Functions

(Note that these are not member functions.)

- DDS_TopicDescription DDS_Topic_TopicDescription (DDS_Topic t)
  This operation 'casts' the provide DDS_Topic to a DDS_TopicDescrition.
- const char * DDS_Topic_get_type_name (DDS_Topic t)
  *This operation returns **type_name** of the provided DDS_Topic.*
- const char * DDS_Topic_get_name (DDS_Topic t)
  *This operation returns **topic name** of the provided DDS_Topic.*
- DDS_DomainParticipant DDS_Topic_get_participant (DDS_Topic t)
  This operation returns DDS_DomainParticipant that owns the DDS_Topic.
- DDS_ReturnCode_t DDS_Topic_enable (DDS_Topic t)
  Enables the DDS_Topic.
- DDS_InstanceHandle_t DDS_Topic_get_instance_handle (DDS_Topic t)
  This operation returns the InstanceHandle_t that identifies the Topic.
- DDS_StatusCondition DDS_Topic_get_statuscondition (DDS_Topic t)
  This operation allows access to the DDS_StatusCondition associated with the Topic.
- DDS_StatusMask DDS_Topic_get_status_changes (DDS_Topic t)
  *This returns the list of **triggered** communication statuses in the Topic.*
- DDS_ReturnCode_t DDS_Topic_set_qos (DDS_Topic t, const DDS_TopicQos *qos)
  Sets the DDS_TopicQos values.
- DDS_ReturnCode_t DDS_Topic_get_qos (DDS_Topic t, DDS_TopicQos *qos)
  *Returns the current DDS_TopicQos settings held in the Topic **t**.*
- DDS_ReturnCode_t DDS_Topic_set_listener (DDS_Topic t, DDS_TopicListener *a_listener, DDS_StatusMask mask)
  *Installs a DDS_TopicListener on Topic **t**.*

- DDS_ReturnCode_t DDS_Topic_set_listener_cd (DDS_Topic t, DDS_TopicListener_cd *a_listener, DDS_StatusMask mask, void *callback_data)

  *Installs a DDS_TopicListener_cd on Topic **t**.*

- DDS_TopicListener * DDS_Topic_get_listener (DDS_Topic t)

  This operation returns the currently installed DDS_TopicListener.

- DDS_TopicListener_cd * DDS_Topic_get_listener_cd (DDS_Topic t)

  This operation returns the currently installed DDS_TopicListener_cd.

- DDS_ReturnCode_t DDS_Topic_get_inconsistent_topic_status (DDS_Topic t, DDS_InconsistentTopicStatus *a_status)

  Provides access to the DDS_InconsistentTopicStatus of the Topic.

3.48.1 Detailed Description

DDS_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 DDS_DomainParticipant_create_topic(). Prior to creating a Topic, the associated data type must be registered with the DomainParticipant via a call to the TypeSupport::register_type() function. [The register_type() function is auto-generated 'type-specific' code.]

3.48.2 Friends And Related Function Documentation

3.48.2.1 DDS_ReturnCode_t DDS_Topic_enable (DDS_Topic *t*) [related]

Enables the DDS_Topic.

A Topic is created either enabled or not based on the DDS_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 DDS_NOT_ENABLED error. DDS_Topic_enable() may be called on an already enabled Topic [it will have no effect].

3.48.2.2 DDS_ReturnCode_t DDS_Topic_get_inconsistent_topic_status (DDS_Topic *t*, DDS_InconsistentTopicStatus * *a_status*) [related]

Provides access to the DDS_InconsistentTopicStatus of the Topic.

As a side-effect, this routine will reset the **total_count_change** status field to zero.

3.48.2.3 DDS_TopicListener * DDS_Topic_get_listener (DDS_Topic *t*) [related]

This operation returns the currently installed DDS_TopicListener.

Note

Because the infrastructure makes a copy of the listener provided in DDS_Topic_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.

3.48.2.4 DDS_TopicListener_cd * DDS_Topic_get_listener_cd (DDS_Topic *t*) [related]

This operation returns the currently installed DDS_TopicListener_cd.

Note

Because the infrastructure holds a pointer to the listener provided in DDS_Topic_set_listener_cd(), the returned structure pointer will match the pointer originally provided. The application should not free the data referenced by the returned pointer.

3.48.2.5 DDS_ReturnCode_t DDS_Topic_get_qos (DDS_Topic *t*, DDS_TopicQos * *qos*) [related]

Returns the current DDS_TopicQos settings held in the Topic **t**.

This routine copies the Topic QoS properites into **qos**.

Note

The qos structure may contain sequences or strings that are populated with dynamic memory. The caller is responsible for freeing the dynamic memory of these items.
For example, the sequence 'qos->topic_data' may have dynamically allocated memory assigned. This can be released by a call to **seq_clear(&qos->topic_data)**.

3.48.2.6 DDS_StatusMask DDS_Topic_get_status_changes (DDS_Topic *t*) [related]

This returns the list of **triggered** communication statuses in the Topic.

If the Topic is not enabled, all statuses will be **untriggered**.

3.48.2.7 DDS_StatusCondition DDS_Topic_get_statuscondition (DDS_Topic *t*) [related]

This operation allows access to the DDS_StatusCondition associated with the Topic.

The returned condition can be added to a DDS_WaitSet.

3.48.2.8 DDS_ReturnCode_t DDS_Topic_set_listener (DDS_Topic *t*, DDS_TopicListener * *a_listener*, DDS_StatusMask *mask*) [related]

Installs a DDS_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.

3.48.2.9 DDS_ReturnCode_t DDS_Topic_set_listener_cd (DDS_Topic *t*, DDS_TopicListener_cd * *a_listener*, DDS_StatusMask *mask*, void * *callback_data*) [related]

Installs a DDS_TopicListener_cd on Topic **t**.

Only one listener may be attached to a Topic at a time. A call to **set_listener_cd()** will replace any current listener with **a_listener**.

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

The infrastructure will not make an internal copy of the listener_cd structure which means that it must be persisted by the application.

3.48.2.10 DDS_ReturnCode_t DDS_Topic_set_qos (DDS_Topic *t*, const DDS_TopicQos * *qos*) [related]

Sets the DDS_TopicQos values.

These QoS values affect the behavior of the DDS_Topic.

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

- dds.h

3.49 DDS_TopicDescription Struct Reference

DDS_TopicDescription is an abstract 'class' that provides the foundation for DDS_Topic, DDS_ContentFilteredTopic, and DDS_MultiTopic.

Related Functions

(Note that these are not member functions.)

- const char * DDS_TopicDescription_get_type_name (DDS_TopicDescription td)
  *This operation returns **type_name** of the provided DDS_TopicDescription.*
- const char * DDS_TopicDescription_get_name (DDS_TopicDescription td)
  *This operation returns **topic name** of the provided DDS_TopicDescription.*
- DDS_DomainParticipant DDS_TopicDescription_get_participant (DDS_TopicDescription td)
  This operation returns DDS_DomainParticipant that owns the DDS_TopicDescription.

3.49.1 Detailed Description

DDS_TopicDescription is an abstract 'class' that provides the foundation for DDS_Topic, DDS_ContentFilteredTopic, and DDS_MultiTopic.

3.49.2 Friends And Related Function Documentation

3.49.2.1 const char * DDS_TopicDescription_get_name (DDS_TopicDescription *td*) [related]

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

The returned string is owned by the **TopicDescription**, do not free it.

3.49.2.2 const char * DDS_TopicDescription_get_type_name (DDS_TopicDescription *td*) [related]

This operation returns **type_name** of the provided DDS_TopicDescription.

The returned string is owned by the **TopicDescription**, do not free it.

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

- dds.h

3.50 DDS_TopicListener Struct Reference

The DDS_TopicListener provides asynchronous notification of DDS_Topic events.

Public Attributes

- void(∗ on_inconsistent_topic)(DDS_Topic the_topic, DDS_InconsistentTopicStatus status)

3.50.1 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().

Note

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

3.50.2 Member Data Documentation

3.50.2.1 void(∗ DDS_TopicListener::on_inconsistent_topic)(DDS_Topic the_topic, DDS_InconsistentTopicStatus status)

on_inconsistent_topic() is 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 struct was generated from the following file:

- dds.h

3.51 DDS_TopicListener_cd Struct Reference

The DDS_TopicListener_cd provides asynchronous notification of DDS_Topic events with additional callback data.

Public Attributes

- void(∗ on_inconsistent_topic)(struct DDS_TopicListener_cd ∗self, DDS_Topic the_topic, DDS_InconsistentTopicStatus status, void ∗callback_data)

3.51.1 Detailed Description

The DDS_TopicListener_cd provides asynchronous notification of DDS_Topic events with additional callback data. This listener can be installed by calling DDS_Topic_set_listener_cd().

3.51.2 Member Data Documentation

3.51.2.1 void(∗ DDS_TopicListener_cd::on_inconsistent_topic)(struct DDS_TopicListener_cd ∗self, DDS_Topic the_topic, DDS_InconsistentTopicStatus status, void ∗callback_data)

on_inconsistent_topic() is 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.

Note

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

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

- dds.h

3.52 DDS_TopicQos Struct Reference

Structure that holds DDS_Topic Quality of Service policies.

Public Attributes

- DDS_TopicDataQosPolicy topic_data

  A sequence of octets associated with a Topic.

- DDS_DurabilityQosPolicy durability

  The durability policy of the Topic.

- DDS_DurabilityServiceQosPolicy durability_service

  Configures the service supporting the TRANSIENT and PERSITENT durability kinds.

- DDS_DeadlineQosPolicy deadline

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

- DDS_LatencyBudgetQosPolicy latency_budget

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

- DDS_LivelinessQosPolicy liveliness

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

- DDS_ReliabilityQosPolicy reliability

  Indicates the level of transport reliability on the Topic.

- DDS_DestinationOrderQosPolicy destination_order

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

- DDS_HistoryQosPolicy history

  The amount of historical data maintained for the Topic.

- DDS_ResourceLimitsQosPolicy resource_limits

  The resource limitations for the Topic.

- DDS_TransportPriorityQosPolicy transport_priority

  The priority to be used for messages on the Topic.

- DDS_LifespanQosPolicy lifespan

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

- DDS_OwnershipQosPolicy ownership

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

3.52.1 Detailed Description

Structure that holds DDS_Topic Quality of Service policies.

See also

- DDS_Topic_set_qos()
- DDS_Topic_get_qos()
- DDS_DomainParticipant_create_topic()
- DDS_DomainParticipant_set_default_topic_qos()

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

- dds.h

3.53 DDS_WaitSet Struct Reference

A DDS_WaitSet maintains a set of DDS_Condition objects and allows the application to wait until one or more of them have a **trigger_value** of TRUE.

Related Functions

(Note that these are not member functions.)

- DDS_WaitSet DDS_WaitSet__alloc ()
  Convenience routine to allocate a DDS_WaitSet.
- void DDS_WaitSet__free (DDS_WaitSet ws)
  Routine to destroy a DDS_WaitSet.
- DDS_ReturnCode_t DDS_WaitSet_wait (DDS_WaitSet ws, DDS_ConditionSeq *active_conditions, DDS_Duration_t *timeout)
  *Causes the controlling thread to block until an attached condition is triggered or **timeout** elapses.*
- DDS_ReturnCode_t DDS_WaitSet_attach_condition (DDS_WaitSet ws, DDS_Condition c)
  *Adds the condition **c** to the WaitSet **ws**.*
- DDS_ReturnCode_t DDS_WaitSet_detach_condition (DDS_WaitSet ws, DDS_Condition c)
  *Removes a condition **c** from WaitSet **ws**.*
- DDS_ReturnCode_t DDS_WaitSet_get_conditions (DDS_WaitSet ws, DDS_ConditionSeq *attached_conditions)
  Retrieves the current list of attached conditions.

3.53.1 Detailed Description

A DDS_WaitSet maintains a set of DDS_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. A WaitSet is created by a call to DDS_WaitSet__alloc(), and destroyed with a call to DDS_WaitSet__free().

3.53.2 Friends And Related Function Documentation

3.53.2.1 DDS_WaitSet DDS_WaitSet__alloc ( ) `[related]`

Convenience routine to allocate a DDS_WaitSet.

Allocates memory which should be returned via DDS_WaitSet__free().

3.53.2.2 void DDS_WaitSet__free (DDS_WaitSet *ws*) [related]

Routine to destroy a DDS_WaitSet.

Releases memory which was previously allocated by DDS_WaitSet__alloc().

Note

The application should call DDS_WaitSet_detach_condition() for each attached Condition prior to calling DDS_WaitSet__free(). [This routine will not destroy any attached Conditions.]

3.53.2.3 DDS_ReturnCode_t DDS_WaitSet_attach_condition (DDS_WaitSet *ws,* DDS_Condition *c*) [related]

Adds the condition **c** to the WaitSet **ws**.

If another thread is currently 'waiting' on the WaitSet, this newly added condition will unblock that thread if its **trigger_value** is TRUE.

3.53.2.4 DDS_ReturnCode_t DDS_WaitSet_detach_condition (DDS_WaitSet *ws,* DDS_Condition *c*) [related]

Removes a condition **c** from WaitSet **ws**.

If the condition is not attached to the WaitSet, the error DDS_RETCODE_PRECONDITION_NOT_MET will be returned.

3.53.2.5 DDS_ReturnCode_t DDS_WaitSet_get_conditions (DDS_WaitSet *ws,* DDS_ConditionSeq * *attached_conditions*) [related]

Retrieves the current list of attached conditions.

Populates the **attached_conditions** sequence. The application is responsible for freeing the memory contained in the sequence.

3.53.2.6 DDS_ReturnCode_t DDS_WaitSet_wait (DDS_WaitSet *ws,* DDS_ConditionSeq * *active_conditions,* DDS_Duration_t * *timeout*) [related]

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.

Note

The application is responsible for clearing the memory contained in **active_conditions** upon return.

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

- dds.h

Chapter 4

Data Structure Index

4.1 Class List

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

Chapter 5

Not Yet Supported

Member DDS_DomainParticipant_create_multitopic This operation is not implemented yet.

Member DDS_DomainParticipant_delete_multitopic This operation is not implemented yet.

Member DDS_DomainParticipant_ignore_publication This operation is not implemented yet.

Member DDS_DomainParticipant_ignore_subscription This operation is not implemented yet.

Member DDS_DomainParticipant_ignore_topic This operation is not implemented yet.

Member DDS_DynamicType_get_string Long Double data types are not supported by CoreDX DDS.

Member DDS_DynamicType_set_string Long Double data types are not supported by CoreDX DDS.

Class DDS_MultiTopic CoreDX does not yet implement MultiTopics.

Member DDS_Publisher_begin_coherent_changes This operation is not yet implemented.

Member DDS_Publisher_end_coherent_changes This operation is not yet implemented.

Member DDS_Publisher_resume_publications This operation is not yet implemented.

Member DDS_Publisher_suspend_publications This operation is not yet implemented.

Class DDS_QueryCondition CoreDX DS does not yet implement QueryConditions as a trigger for a WaitSet.

Index